VLSI architectures design for encoders of High Efficiency Video Coding (HEVC) standard

POLITECNICO DI TORINO

SCUOLA DI DOTTORATO

Ph.D. in Electronics and Communications Engineering – XXVIII cycle

Ph.D. Dissertation

VLSI architectures design for encoders of High Efficiency Video Coding (HEVC) standard

Guoping Xiao

Supervisor
Prof. Guido Masera
Prof. Maurizio Martina

Coordinator of the Ph.D. course
Prof. Ivo Montrosset

May 2016

To my parents, Zhao Heyun and Xiao Caiheng, and my beloved fiancee, Liu Yu

Summary

The growing popularity of high resolution video and the continuously increasing demands for high quality video on mobile devices are producing stronger needs for more efficient video encoder. Concerning these desires, HEVC, a newest video coding standard, has been developed by a joint team formed by ISO/IEO MPEG and ITU/T VCEG. Its design goal is to achieve a 50% compression gain over its predecessor H.264 with an equal or even higher perceptual video quality. Motion Estimation (ME) being as one of the most critical module in video coding contributes almost 50%-70% of computational complexity in the video encoder. This high consumption of the computational resources puts a limit on the performance of encoders, especially for full HD or ultra HD videos, in terms of coding speed, bit-rate and video quality. Thus the major part of this work concentrates on the computational complexity reduction and improvement of timing performance of motion estimation algorithms for HEVC standard.

First, a new strategy to calculate the SAD (Sum of Absolute Difference) for motion estimation is designed based on the statistics on property of pixel data of video sequences. This statistics demonstrates the size relationship between the sum of two sets of pixels has a determined connection with the distribution of the size relationship between individual pixels from the two sets. Taking the advantage of this observation, only a small proportion of pixels is necessary to be involved in the SAD calculation. Simulations show that the amount of computations required in the full search algorithm is reduced by about 58% on average and up to 70% in the best case.

Secondly, from the scope of parallelization an enhanced TZ search for HEVC is proposed using novel schemes of multiple MVPs (motion vector predictor) and shared MVP. Specifically, resorting to multiple MVPs the initial search process is performed in parallel at multiple search centers, and the ME processing engine for PUs within one CU are parallelized based on the MVP sharing scheme on CU (coding unit) level. Moreover, the SAD module for ME engine is also parallelly implemented for PU size of 32×32. Experiments indicate it achieves an appreciable improvement on the throughput and coding efficiency

of the HEVC video encoder.

In addition, the other part of this thesis is contributed to the VLSI architecture design for finding the first W maximum/minimum values targeting towards high speed and low hardware cost. The architecture based on the novel bit-wise AND scheme has only half of the area of the best reference solution and its critical path delay is comparable with other implementations. While the FPCG (full parallel comparison grid) architecture, which utilizes the optimized comparator-based structure, achieves 3.6 times faster on average on the speed and even 5.2 times faster at best comparing with the reference architectures. Finally the architecture using the partial sorting strategy reaches a good balance on the timing performance and area, which has a slightly lower or comparable speed with FPCG architecture and a acceptable hardware cost.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Prof. Guido Masera and Prof. Maurizio Martina. Attributed to their professional guidance, kindly encouragement and great patience I can finally accomplish the thesis work.

I am deeply grateful to my fiancee for her careful attendance and strong support all the way.

I also appreciate the help and encouragement from many of my friends, especially Zhang Qiutao, Bai Ou and Du Boyang.

I am sincerely grateful to my parents and elder sisters for their selfless love and support.

Finally I sincerely thank all those who have ever helped me these years, without your support, I would never reach where I am now.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As smart phone, tablet and HDTV has been an inseparable part of our life, digital video plays a more and more important role in the modern society. With the growing popularity of high resolution and high quality videos, digital video compression techniques has a increasing significance in the telecommunication and multimedia system in which bandwidth is still a valuable resource.

Digital video signal is represented as a sequence of real-world or virtual visual scenes [5, 6]. Each frame is composed of a two-dimensional grid of pixels. The product of the number of columns and rows of pixels in the grid is known as resolution of the frame or video. For instance, a frame with the resolution of standard definition (SD) has 720×480 pixels. Whereas its resolution can be various as small as 176×144 (QCIF), or as large as 1920×1080 (FHD) [7] or even 4K and 8K UHD. A pixel generally consists of three fundamental components represented in RGB or YUV. Each component normally takes 8 or 12 digital bits. It can be observed that the size and data rate of a raw video signal is tremendous. Taking a FHD video (typically with a frame rate of 30 fps) as an example, its data rate is 30×1920×1080×3×8 = 1.49 Gbit per second. Apparently it has already far surpassed the capacity of today's telecommunication infrastructure. Moreover, situation will be even more critical if targeting for the higher resolution videos like UHD.

1.1 General Video Coding System

Since video sequences are sampled spatially and temporally with a high frame rate (e.g. 30 fps). There exists an immense amount of temporal and spatial redundant information since both the consecutive frames and the blocks inside one frame are highly correlated. Digital video coding, as the process of compression and decompression of video signal, generally is to exploit and eliminate these redundancies from the video sequences in order to alleviate the requirement on bandwidth for transmission and space for storage [8].

1.1.1 Basics of Digital Video Coding

Overview

Generally, a raw video is firstly captured by a camera or video recording device in digital or analogue format [9]. Figure.1.1 illustrates the block diagram of a typical digital video coding system. The pre-processing unit is to pre-process the raw video signal, in which if the format of raw videos from CCD camera is analogue, it will be firstly digitized into RGB images and then the digital signal in RGB presentation is converted into YUV representation [10]. In the next the signals will be compressed and encoded by the encoding unit. This encoding process is able to compress the raw video in a high compression ratio and maintains an good video quality through powerful smart strategies and high efficient algorithms. On the contrary, the output video for displaying can be acquired through decoding and post-processing modules as shown in Figure.1.1.

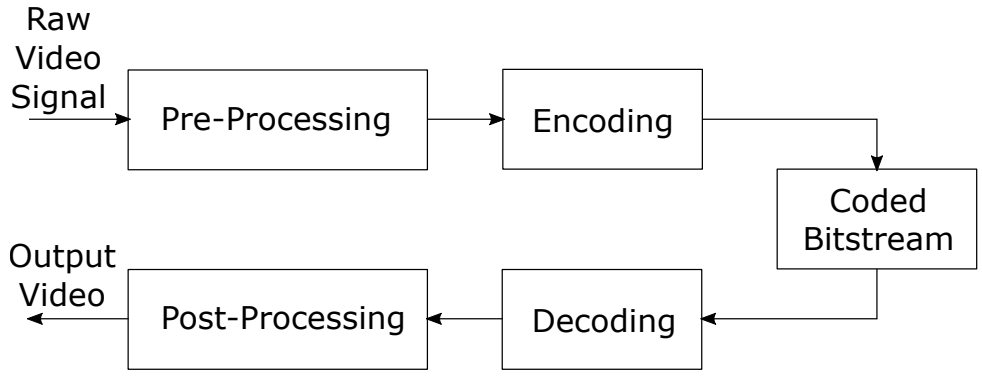


Figure 1.1: Block diagram of a general video coding system.

Concepts and Definitions

As aforementioned video compression seeks to eliminate redundancies from video sequences, there are two types of compression : lossless compression and lossy compression, which are categorized depending on whether the encoding process brings about loss in the information after compression. Although the lossy compression can't fully reconstruct the video sequences thus suffers from a degradation of video quality, it achieves a compression ratio almost 95% higher than lossless compression. Therefore lossy compression is utilized by most practical video encoding systems. Before diving into the details of compression techniques, some basic concepts and definitions of video coding are introduced firstly in the following.

Color Space- Generally speaking, each sample point of the colorful video signal displayed by most of the digital video applications requires three numbers to accurately represent a color. One popular color space system is the RGB color space, of which the

three numbers indicate the relative proportions of Red, Green and Blue. Each number is represented with 'N' digital bits, for instance, a number of 8-bit can hold 256 color levels for each color element. A Combination of these three color elements in varying proportions can produce any color. Whereas for the sake of decreasing the bandwidth requirement for transmitting video signals, the YCrCb color space is widely utilized because normally it can represents every pixel with less bits. YCrCb employs a luminance/chrominance coordinate which extracts the luminance from the color information and represent luma (Y) part using a higher resolution than chroma color parts (Cr and Cb) [11]. Since according to human visual system (HVS) it is more sensitive to luminance than chromince [12]. In this way YCrCb can represents a color image more efficiently. The converting relationship between the two color spaces, RGB and YCrCb, is expressed by

$$\begin{aligned} Y &= k_r R + k_g G + k_b B \\ Cr &= R - Y \\ Cg &= G - Y \\ Cb &= B - Y \end{aligned} \tag{1.1}$$

where k_r, k_g, k_b are weighting factors. It is easy to observe that Cr+Cg+Cb is a constant, thus using only two of them is sufficient for storage or transmission and the last component can be obtained from the other two. Furthermore, the YCrCb color space and its variants are referred as YUV. Based on several typical weighting factors there are three common YCrCb sampling patterns widely supported by video coding standards which are shown in Figure.1.2. The 4:4:4 sampling indicates Y, Cr and Cb components all have full resolution

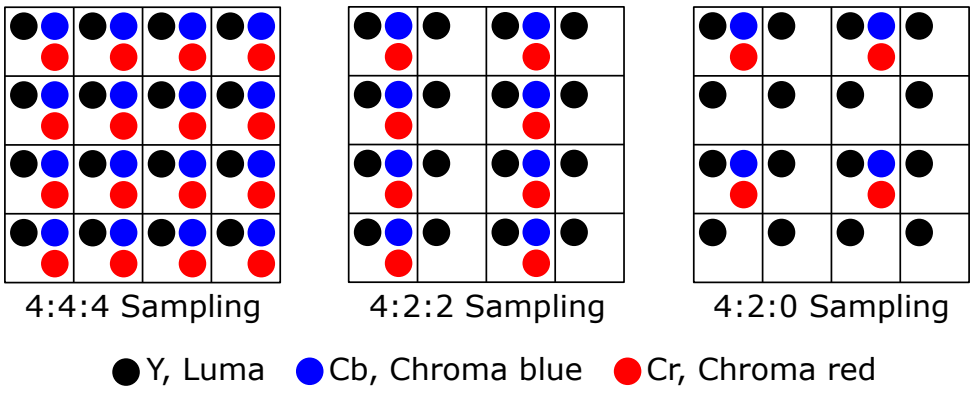


Figure 1.2: Three typical YCrCb sampling patterns.

so that it retains full fidelity of the chrominance component. In the 4:2:2 sampling, chrominance components are sub-sampled into half resolution in horizontal direction. While 4:2:0 sampling half the samples of chrominance components in both horizontal and vertical resolution. Since in 4:2:0 sampling each of Cr and Cb components only holds 1/4

of the number of samples of Y component, video in 4:2:0 YCrCb format needs half of the storage space or transmission bandwidth relative to that of 4:4:4 YCrCb video. As a result, 4:2:0 sampling is the most widely utilized one.

Frame Rate- As mentioned before, a video is captured by camera temporally at a specific frequency. The frame rate represented in Hz or frames per second (fps) is referred as the rate for capturing or playing back. Typically the Standard Definition television supports a frame rate of 25 or 30 fps. Whereas in order to generate a video with very smooth apparent motion, it requires 50 or 60 frames per second or even higher like 120 fps. However, high frame rate means a very high data rate. Thus choosing a reasonable frame rate is very important when encoding a video signal for specific applications.

Video Resolutions- Typically the video resolution is expressed as the size of frame, namely length × width, in pixels. There are some common resolutions including:

8K UHD (Ultra High Definition): it refers to a resolution of 7680×4320.

4K UHD: it refers to a resolution of 3840×2160.

FHD (Full High Definition): it refers to a resolution of 1920×1080.

HD: it normally refers to a resolution of 1280×720.

SD (Standard Definition): it refers to a resolution of 640×480.

CIF (Common International Format): it refers to a resolution of 352×288.

QCIF (Quarter CIF): it refers to a resolution of 176×144.

Color Depth- It is also called as bit depth, which is usually referred to the number of bits used for one pixel or one color component [1]. Color depth defines how many color levels can be expressed. Typically a 8-bit color depth enables 256 kinds of colors which is widely utilized by monitor of consumer devices. Moreover, for some special applications there are 24-bit color depth for true color and 30/36/48-bit color depth for deep color [13].

1.1.2 Hybrid Video Coding

Resorting to bunch of compression tools, hybrid coding system regularly consists of several standard video compression techniques [14–19]. Figure 1.3 depicts a typical hybrid video coding system, which is composed of the encoding part and decoding part. As aforementioned, the main purpose of video coding is to compress the video by reducing the redundancy of the video sequences. In general, there are two kinds of redundancy existing in video signal: the spatial and the temporal. The spatial redundancy is exploited through intra-frame prediction, of which the mechanism is to find a best matching block

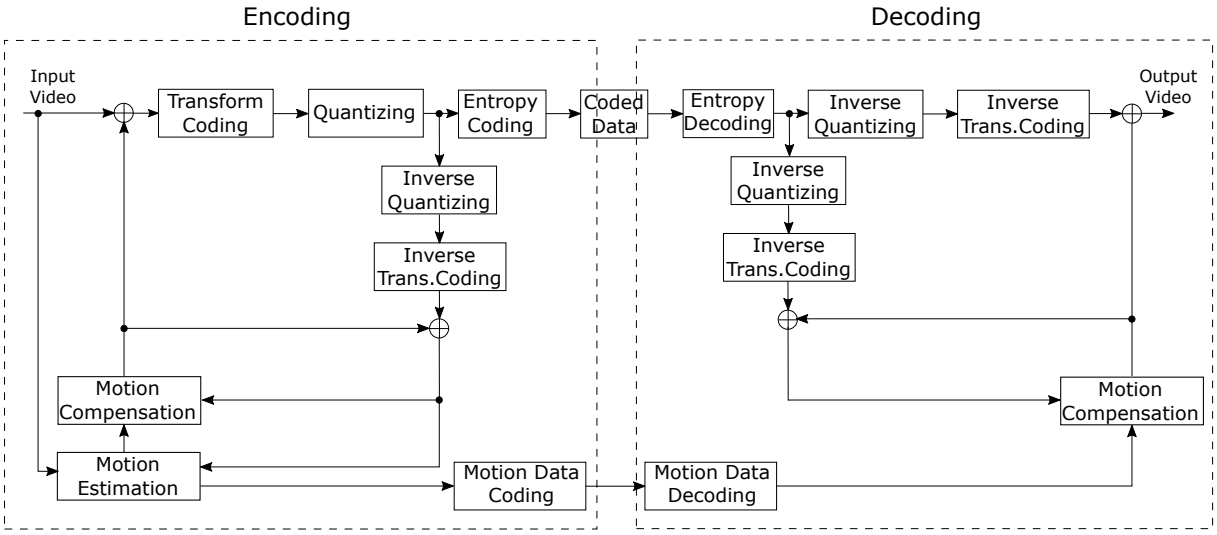


Figure 1.3: Block diagram of general hybrid video coding system.

for current one from its left and upper neighboring blocks assuming a coding mode of raster-scan order is applied. It is because there is a high correlation between neighboring pixels inside a frame. As a result in intra-frame prediction a block is reconstructed by the blocks that has already been encoded and stored in the bit-stream for output. While on the contrary inter-frame prediction is designed to deal with the temporal redundancy existed between consecutive frames such as the frame in the past or in the future. The temporal redundancy is normally referred to the correlation of two adjacent frames, since they usually have similar backgrounds or a same object with a relative motion. Thus the purpose of inter-frame prediction is to search for a best matching block for the current one in the reference frame from the past or the future or both of them.

After the best matching candidate for the current block is found by either intra-frame or inter-frame prediction, only the difference between the original block and the predicted one needs to be encoded. As seen from Figure 1.3, these differences (also called residuals) will be further processed and converted into the coefficients of transform domain by the Transform Coding Unit. Transforming is an operation to reconstruct the residuals into signals of frequency domain using transform algorithms such as Discrete Cosine Transform (DCT), which will make the signals more compact. Therefore more redundant information can be eliminated through transforming.

In the next the transforming coefficients will be further quantized by quantization operation which is trying to reduce the range of the signal for saving the number of bits for transmission or storage. And there is an important criterion measured by Quantization Parameter (QP) for the step size between consecutive re-scaled values [2]. Since a large QP can reach a high compression ratio but result in a relative low video quality because of a rough approximation of original signal. On the contrary a small QP will realize a

more closely approximation of original signal, so it will achieve high video quality but degrade the compression efficiency. Finally the quantized transform coefficients together with the motion data are converted into compressed output bit-stream for transmission by the Entropy Encoding Unit shown in Figure 1.3. On the other hand the decoding part performs the inverse operations on the bit-stream to reconstruct the video for playing.

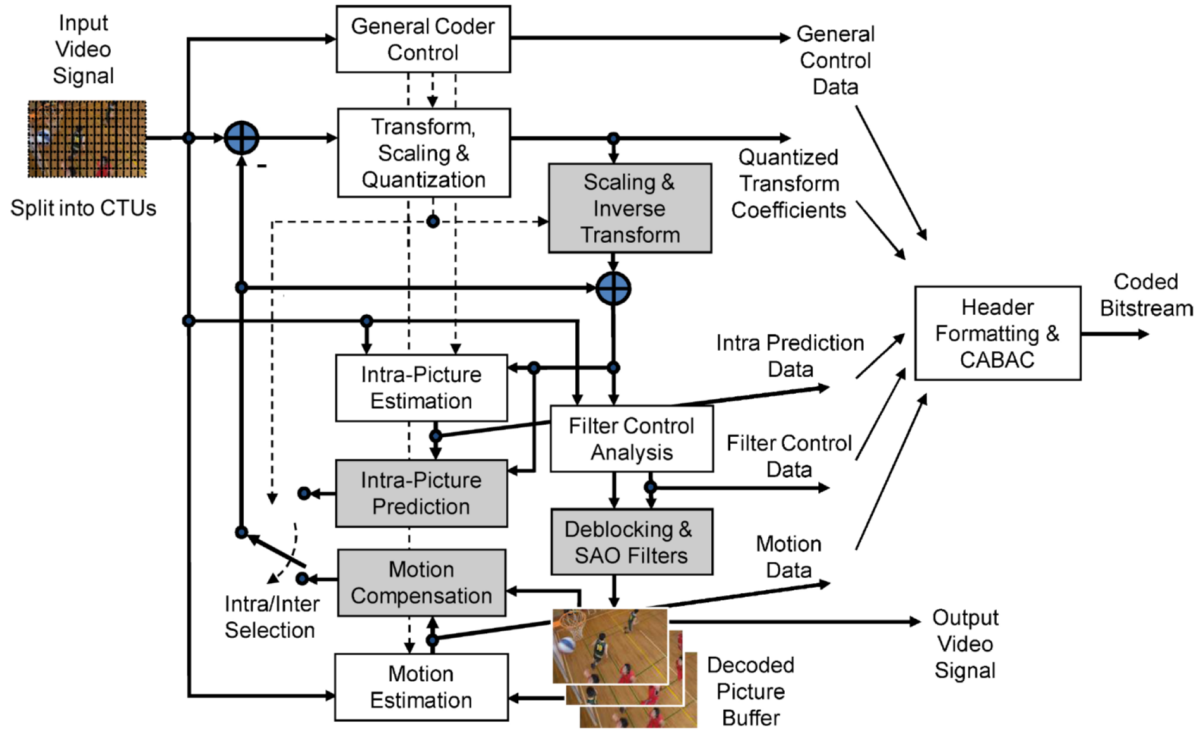


Figure 1.4: Block diagram of HEVC coding system [1].

1.2 Overview of Current Video Coding Standard: HEVC

The High Efficiency Video Coding standard is formally issued by the Joint Collaborative Team on Video Coding (JCT-VC) founded by the international standardization organizations of ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG) [20]. Besides addressing all the available application of the preceding video coding standard, H.264/AVC, two critical aspects: higher and higher video resolution and more and more parallelized architectures for coding are the main concerning in the design of HEVC [1]. With respect to H.264, HEVC aims to gain an 50% more compression with no loss on the video quality. Being the same with its predecessors such as H.264, H.263, MPEG-2 and so on, HEVC standard defines the syntax and structure for compressing videos and generating decoded sequences, and specifies a bunch of necessary techniques or tools for video coding. A typical block diagram of HEVC coding system

is depicted in Figure 1.4. In the following sections new features of HEVC standard are introduced in details.

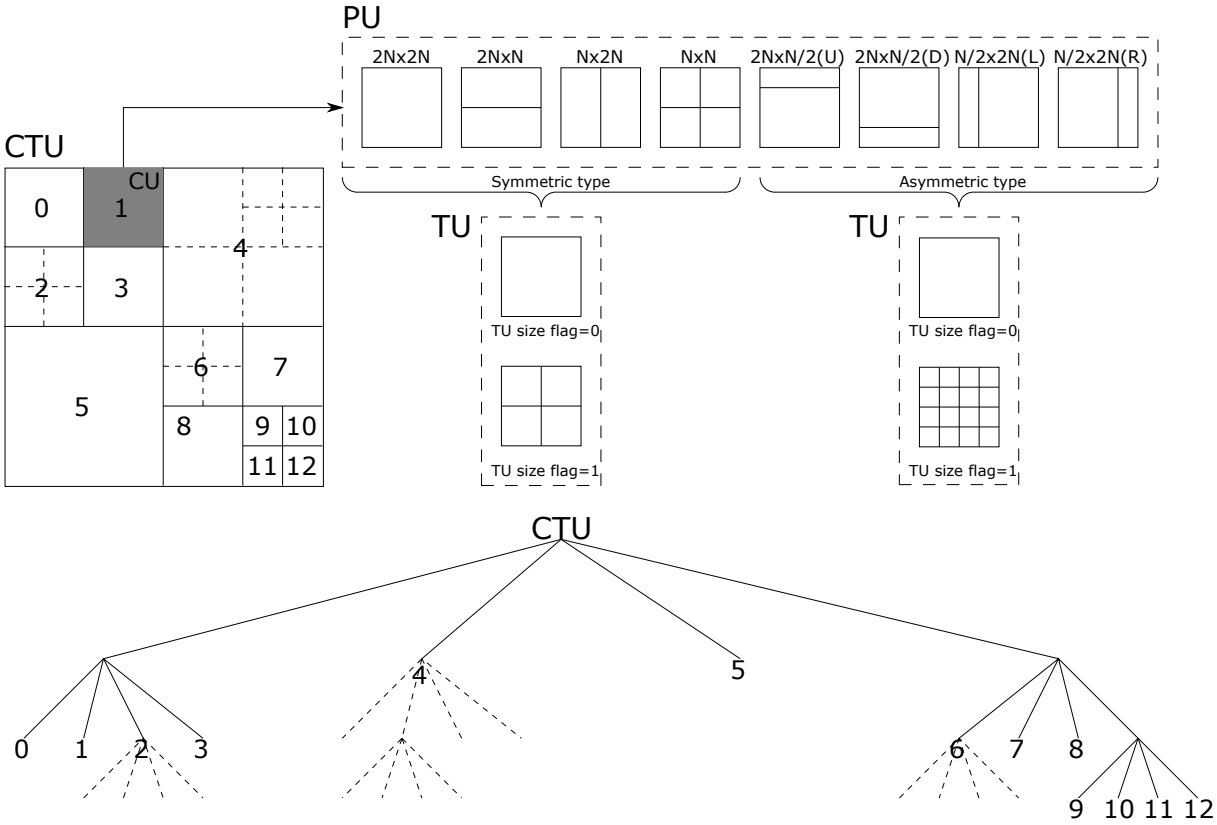


Figure 1.5: HEVC quad-tree partitioning structure including CU, PU and TU (solid line for CU, dashed line for TU).

1.2.1 Partitioning

In the former video coding standard H.264 variable sizes of Macro-Block (MB) from 4×4 to as large as 16×16 are supported [21]. Whereas larger MB size is utilized in HEVC standard, which extends the maximum traditional MB size of 16×16 in H.264 to 64×64 for facilitating the compression of high definition videos. Additionally more flexible partitioning of video frames is supported for improving the coding efficiency, where available block size varies from 4×4 up to 64×64 including symmetric partitioning such as $2N \times 2N$, $2N \times N$, $N \times 2N$ and $N \times N$, and also asymmetric motion partitioning (AMP) for instance $2N \times N/4(U)$, $2N \times N/4(D)$, $N/4 \times 2N(L)$ and $N/4 \times 2N(R)$. In particular $N \times N$ is only allowed for minimum coding unit (CU) and the AMP is not applied to CU whose size is smaller than 16×16. Figure 1.5 illustrates the partitionings and quad-tree structure utilized in the HEVC standard, in which the analogous structure, called coding tree unit (CTU), is spitted into CUs using a quad-tree partitioning structure, and

a CU can be further sub-divided into prediction units (PU) for inter-frame or intra-frame prediction and its transformation is performed using one or more transform units (TU).

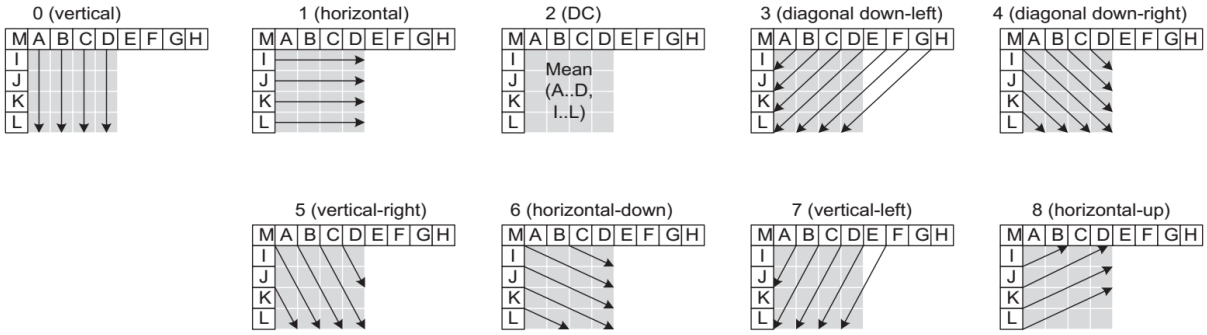


Figure 1.6: Intra prediction modes for 4×4 block size in H.264 [2].

1.2.2 Intra-frame Prediction

According to the syntax of HEVC standard, the choice of intra-frame or inter-frame prediction of a PU is made at the CU level. When the intra mode is determined for the CU, as discussed in Section 1.1.2, intra-frame prediction is to code PUs of the CU using its upper and left neighboring PUs which have already been encoded. In H.264 standard, 9 prediction modes are available for performing the prediction of 8×8 and 4×4 block size, that is shown in Figure 1.6 for 4×4 block size. And for block size 16×16 there are only 4 prediction modes available: vertical, horizontal, DC and plane. Whereas in HEVC standard for the sake of improving prediction accuracy, the total number of available prediction modes is increased to 35, 33 angular modes as well as the DC and planar modes, which is depicted in Figure 1.7. Specifically the number of most probable intra prediction modes for every PU is set to 3 which is determined by the modes of two neighboring PUs: the one on the left and the one on the upper.

1.2.3 Inter-frame Prediction

Since only the correlation between neighboring blocks inside a single frame is exploited by intra-frame prediction, it can not reach a high compression ratio. Most of pictures in the video sequence is encoded using inter-frame prediction because it reduces much more redundancy by employing the temporal correlation between blocks from different frames. With respect to H.264 standard, in HEVC the advanced motion vector prediction (AMVP) technique utilizing both spatial and temporal motion vectors has improved the accuracy of motion vector prediction. Moreover, with the replacement of spatial direct mode in H.264, a new technique of block merging mode is applied in the inter-frame

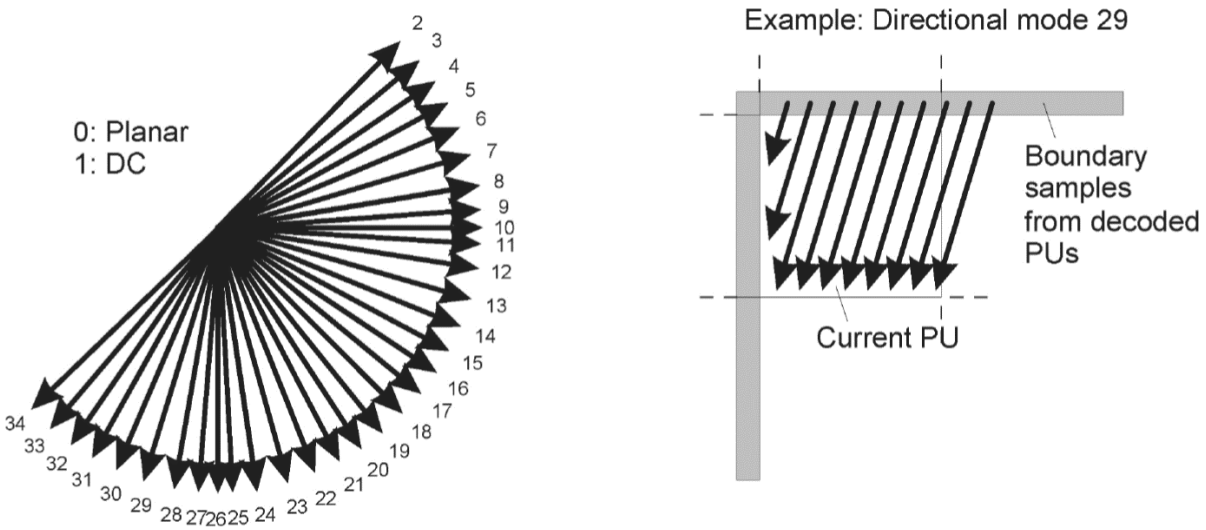


Figure 1.7: Intra prediction modes in HEVC standard [1].

prediction of HEVC, which further enhance the bit-rate reduction by joining the blocks with equal motion parameters and sharing the motion data. An example for block merging is presented in Figure 1.8, in which the shared motion information is held by the seed block (S) and the other merged blocks (M) copy it from S when required. As for the fractional inter-frame prediction, in order to realize high precision interpolation a 8-tap and 7-tap interpolation filter are utilized for the half-pixel positions and quarter-pixel positions respectively [22].

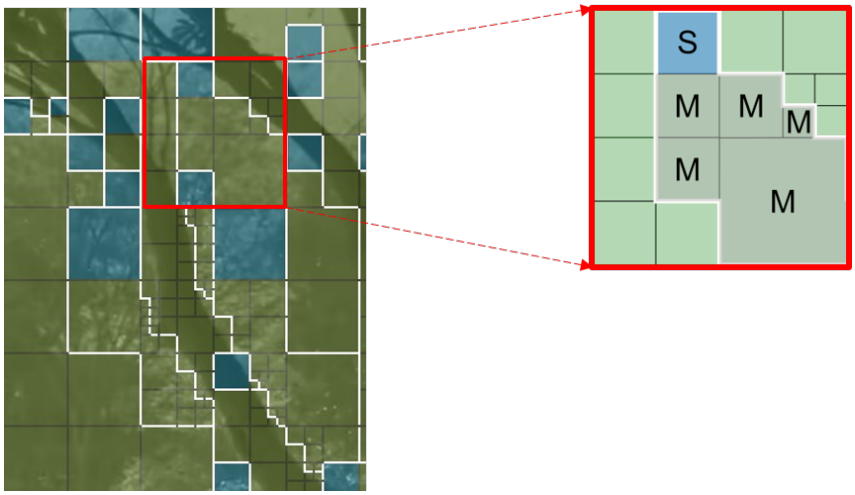


Figure 1.8: An example for block merging in HEVC [3].

1.2.4 Transform and Quantization

After the motion estimation, all the prediction error residuals are transformed into a set of coefficients for efficient transmission and storage. In HEVC standard, as indicated in Figure 1.5, TU size of 4×4, 8×8, 16×16 and 32×32 are supported. The 2-D transforms based on discrete cosine transform (DCT) are designed for them and special efforts are particularly spent on selecting the value of the transfrom matrix for retaining the property of easy-to-implementation [1]. Furthermore when conducting the transform for block size of 4×4 in intra-frame prediction mode, an alternative integer transform grounded on discrete sine transform (DST) is available for use. As for the quantization, a same quantizing mechanism with H.264 is applied with a QP value varying from 0 to 51.

1.2.5 Entropy Coding

In HEVC standard a bit-stream is produced using motion parameters, prediction modes, quadtree partitioning information, quantized transform coefficients and some other control data through entropy coding. And only one entropy coding method, Context-Adaptive Binary Arithmetic Coding (CABAC), is specified in the standard. Although there is no change made on the core algorithm of CABAC, it is optimized on the aspects of context modeling, adaptive coefficient scanning, coefficient coding, sign data hiding and so on to improve its throughput.

1.3 Motion Estimation Algorithms

Motion Estimation (ME) as one of the most significant part of video coding contributes a great proportion of the compression and also occupies a large part of the computational complexity of the encoder [23–26]. There are a lot of algorithms proposed to conduct the ME operation [24,27–46]. Full Search is a ME strategy with a highest estimation accuracy which exhaustively compares all available candidate blocks in the search area with the current one to find the best matching block, thus it is at a high computational cost. With respect to Full Search, many fast search algorithms for ME are developed by reducing the number of search points to achieve an improvement on complexity degradation and throughput but at a cost of video quality loss. In the following, details of some well known fast search algorithms such as Three Step Search (TSS) [44] and Diamond Search (DS) [45, 46] are introduced.

1.3.1 Three Step Search

The TSS starts the search at the motion vector predictor (MVP), namely search center, assuming its coordinate is (MVP_x, MVP_y). Thus with a step size of d, the coordinates of the 9 search positions for each step of TSS are (MVP_x, MVP_y), ($\mathrm{MVP}_x - d$, $\mathrm{MVP}_y - d$), (MVP_x, $\mathrm{MVP}_y - d$), ($\mathrm{MVP}_x + d$, $\mathrm{MVP}_y - d$), ($\mathrm{MVP}_x + d$, MVP_y), ($\mathrm{MVP}_x + d$, $\mathrm{MVP}_y + d$), (MVP_x, $\mathrm{MVP}_y + d$), ($\mathrm{MVP}_x - d$, $\mathrm{MVP}_y + d$) and ($\mathrm{MVP}_x - d$, MVP_y). The best matching position of current step will be the search center of next step and the step size d is updated with a decrement. This algorithm is terminated after exactly 3 steps with the position of minimum distortion in the third step as the final best match. An example with a initial step size of 3 and a decrement of 1 in each step is shown in Figure 1.9, where gray circles represent best match position in each step and the one in third step marks the final best match.

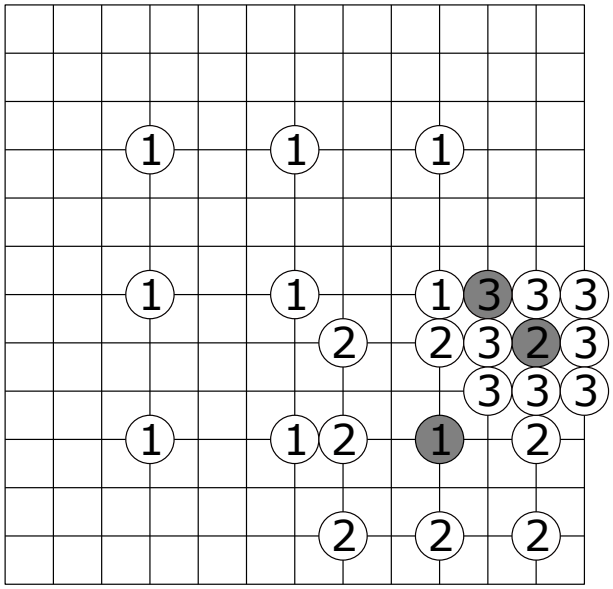


Figure 1.9: An example of Three Step Search.

1.3.2 Diamond Search

Although there are also 9 search positions initially in DS algorithm, these positions are placed according the shape of a diamond at the center of MVP. It is not as same as a square pattern in TSS algorithm. Figure 1.10 exhibits an instance of DS algorithm. In every round of search, if the point with minimum distortion is on the vertex or face of the diamond pattern, DS will continue to do the next search round at the new center. From Figure 1.10 we can see except for the first round, 3 to 5 new positions will be evaluated in each following round. The termination criterion of DS algorithm is that it will not stop

searching until the best matching position is at the center of diamond pattern.

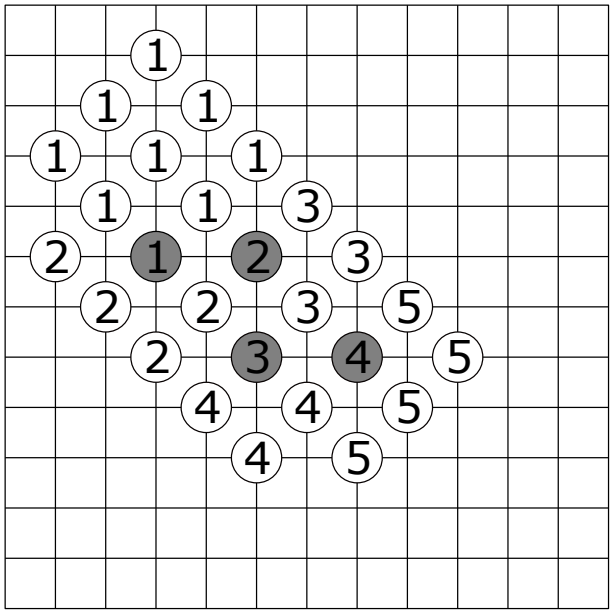


Figure 1.10: An example of Diamond Search.

TSS and DS algorithms or their enhanced variants have been widely used. In the HEVC reference software, the Test Zone (TZ) search is adopted as the default fast search algorithm, which is also based on the DS search mechanism.

1.4 Major Contributions

In this work, research efforts are dedicated to the computational complexity reduction and improvement of timing performance of motion estimation algorithms for HEVC and the implementation and optimization of their VLSI architectures.

Firstly, relative to the conventional computing method of SAD (Sum of Absolute Difference) a new strategy to calculate the SAD is designed grounded on the statistics on property of pixel data of video sequences. With this strategy the amount of computations required in the full search algorithm is greatly reduced, thus the total encoding time is saved by a large scale. Moreover, in order to strengthen the performance of TZ search for HEVC, a novel scheme of parallel initial search is applied utilizing multiple motion vector predictors. Besides, with the mechanism of sharing MVP on the CU level the ME engines for PUs inside one CU are parallelized. Applying with these enhancement, the proposed TZ search algorithm achieves an appreciable improvement on the throughput and coding efficiency of the HEVC video encoder. Finally VLSI architectures based on the schemes of BWA (Bit-Wise AND) and parallel comparison for finding the first W

maximum/minimum values are developed to degrade the hardware cost and improve the timing performance.

1.5 Organization of the Dissertation

Chapter 2 gives an overview of distortion measure criterion in ME and introduces the new SAD computing strategy for full search algorithm.

Chapter 3 firstly places an introduction on the original TZ search algorithm from HEVC reference software, and in the next presents several proposed schemes for enhancing the TZ search and its detailed hardware implementation.

Chapter 4 presents two kinds of VLSI architectures for finding the first W maximum/minimum values based on BWA (Bit-Wise AND) and parallel comparison strategies respectively. The performance of these architectures and comparisons with reference architectures are also introduced.

Chapter 5 addresses a summary for this research work. The ideas and directions for future work are also discussed.

Chapter 2

A New SAD Computing Algorithm for Full Search Motion Estimation

Motion Estimation (ME) is one of the most critical module in video encoder. It occupies almost 50%-70% of computational complexity of codecs [23–26], which will be a bottleneck for further improving the performance of encoders in terms of coding speed, bit-rate and video quality when targeting for full HD or ultra HD videos. Therefore improving the efficiency and reducing the computational load of ME component without degrading video quality is of great significance. In this chapter, based on the statistics on property of pixel data of video sequences, a new strategy to calculate the SAD (Sum of Absolute Difference) for ME is explored. Before introducing the details of this strategy, an discussion on the ME distortion measure criterion is given at first.

2.1 ME Distortion Measure Criterion

As mentioned in section 1.3, the main task of ME is to find the best matching block within the search window in the reference frame for the current block and generate the best motion vector. Figure.2.1 shows an example for block matching ME. In order to find the best matching block (highlighted by green box), it is necessary to search within the whole search window (highlighted by blue box) for the most similar one. Thus a matching criterion or distortion computing method is required to measure the similarity between a possible candidate block and the current block. However, the perceptual similarity or distortion in visual content is very difficult to quantify, since human visual system is complex and hard to understood [47]. In practice, for the digital video signal there are many distortion measure models [48, 49]. Amongst, Mean Squared Error (MSE) and Sum of Absolute Difference (SAD) are in the most common use. We assume $f(i, j, t)$ represents the intensity of a pixel at the coordinate (i, j) in the frame t and accordingly

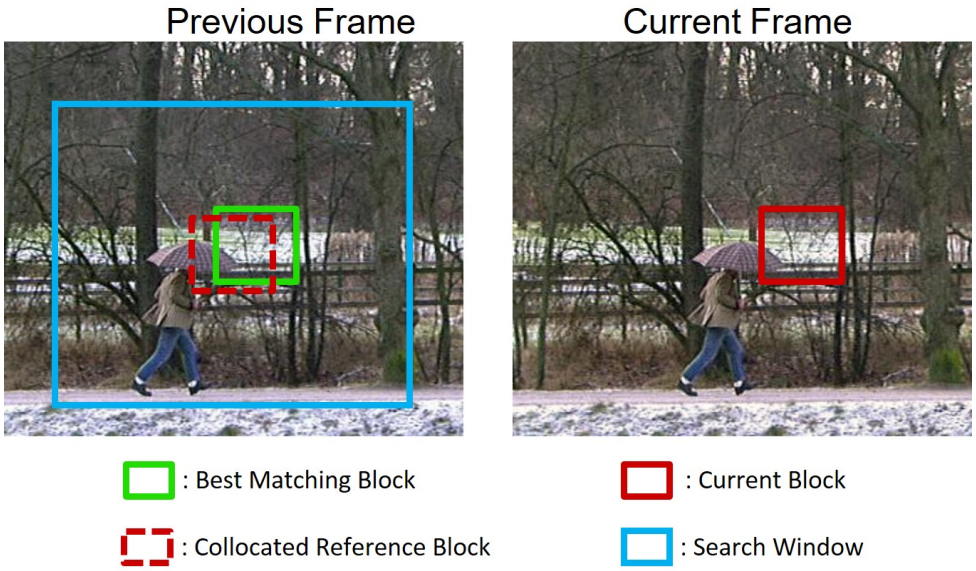


Figure 2.1: An example for block matching ME.

$f(i + MV_x, j + MV_y, t - 1)$ is the intensity of a pixel of a candidate block in the $t - 1$ frame as a reference with a motion vector (MV_x, MV_y), furthermore, the size of a target MB is $N \times N$ pixels, and search range is of $\pm W$ in each direction, so the search window is of size $(2W + N) \times (2W + N)$. Supposing current block is at the location of (u, v), then MSE distortion measure model [50] is defined as

$$MSE(MV_x, MV_y) = \frac{1}{N \times N} \sum_{i=u}^{u+N-1} \sum_{j=v}^{v+N-1} [f(i, j, t) - f(i + MV_x, j + MV_y, t - 1)]^2. \quad (2.1)$$

From Equation 2.1 it is clear that MSE calculates the square of the difference between every two corresponding pixels of current and candidate block, then sums them up and computes its average as the final value. As a result, this value is used to assess the similarity, or the matching degree, of a candidate block. Thus in order to obtain the best matching block, the MSE model is applied for every candidate block with a different motion vector (MV_x, MV_y) within the search window defined by the search range in the reference frame. Finally a candidate block with the minimum MSE will be denoted as the best matching block which is given by

$$MSE_{min} = min\{MSE(MV_x, MV_y)\}, \quad (-W, -W) \leq (MV_x, MV_y) \leq (+W, +W). \quad (2.2)$$

The MSE can be interpreted as the Euclidian distance [51] between the candidate block and target block which is considered as a good block similarity measure model, because it is very close to the perceived similarity by the human visual system. Although MSE

can achieve a precise distortion assessment, it is at the expense of high computational complexity. From Equation 2.1 for one $N \times N$ candidate block, it spends N^2 subtractions, N^2 squaring operations (or multiplication), and N^2-1 additions on calculating the MSE result. Whereas considering the HEVC standard is targeting for FHD or UHD videos and its search range is enlarged to ±64, the MSE will result in tremendous computations which is extremely not practical.

On the meantime, there is a similar block distortion measure model, Sum of Absolute Difference (SAD) [52], which calculates the absolute difference between every two pixels of current block and candidate block instead of the square difference in MSE. The SAD at location (u, v) with a motion vector $f(i + MV_x, j + MV_y, t - 1)$ is denoted as

$$SAD(MV_x, MV_y) = \sum_{i=u}^{u+N-1} \sum_{j=v}^{v+N-1} |f(i, j, t) - f(i + MV_x, j + MV_y, t - 1)|, \tag{2.3}$$

and similarly the best matching block is with the minimum SAD given by

$$SAD_{min} = min\{SAD(MV_x, MV_y)\}, \quad (-W, -W) \leq (MV_x, MV_y) \leq (+W, +W). \tag{2.4}$$

From aforementioned equations the SAD distortion measure model needs N^2 subtractions with absolute operation and N^2-1 additions for every candidate block. Compared to MSE, with the absence of multiplication operations to get the distortion results, it is more feasible for the hardware implementation despite of less accuracy.

2.2 The New SAD Computing Algorithm

2.2.1 A New Strategy to Calculate the SAD

As discussed in the beginning of this chapter, in the video coding system the ME module is one of the most critical part which contributes almost 50%-70% the computational complexity of the video encoder. Therefore it is highly meaningful to design more efficient algorithm and architecture to reduce the amount of computations of ME. Although the SAD criterion is better than the MSE on the complexity, it still requires a huge amount of computations since it involves every pixel of the target block and perform the subtracting, absoluting and adding operations. Especially in the full search algorithm for the high definition videos, the situation will be even worse. Based on the algorithm presented in [4], a new SAD computing algorithm is proposed to improve the efficiency of the SAD architecture based on the statistics on property of pixel data of video sequences.

Let Ψ and Φ be the two sets of N natural numbers denoted as:

$$\Psi = \{\psi_0, \psi_1, \psi_2, ..., \psi_i, ..., \psi_{N-2}, \psi_{N-1}\} \tag{2.5}$$

and

$$\Phi = \{\varphi_0, \varphi_1, \varphi_2, ..., \varphi_i, ..., \varphi_{N-2}, \varphi_{N-1}\}. \tag{2.6}$$

If we sum up all the elements of Ψ and Φ respectively using

$$\Psi_{sum} = \sum_{i=0}^{N-1} \psi_i \tag{2.7}$$

and

$$\Phi_{sum} = \sum_{i=0}^{N-1} \varphi_i. \tag{2.8}$$

Since targeting for the data processing for video signals, we assume that $0 \leq \psi_i,\ \varphi_i \leq 255$, $\forall \psi_i \in \Psi$ and $\forall \varphi_i \in \Phi$. Based on the mathematical statistics, it is true that

$$\mathbb{N} > \mathbb{M} \quad if\ \Psi_{sum} > \Phi_{sum} \tag{2.9}$$

and vice versa, where $\mathbb{N}$ denotes the number of elements that $\psi_i > \varphi_i$(i$\in$(0, N-1)) and similarly $\mathbb{M}$ denotes the number of elements that $\varphi_i > \psi_i$(i$\in$(0, N-1)). Thereby according to Equation 2.9, if the sum of set Ψ is larger than that of Φ, based on the one-to-one correspondence the number of elements that ψ_i is smaller than φ_i is much less than the opposite case. In that case in order to calculate the SAD between Ψ and Φ, the sum difference is firstly obtained by

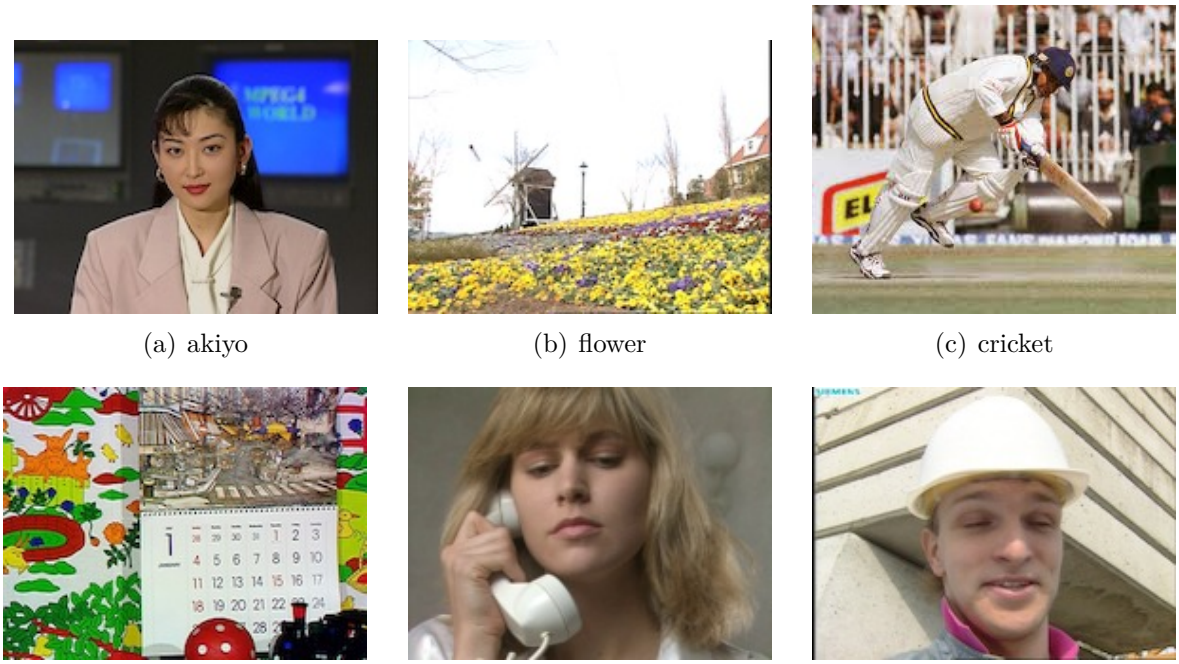
$$\text{Diff}_{sum} = \begin{cases} \Psi_{sum} - \Phi_{sum} & if\ \Psi_{sum} > \Phi_{sum} \\ \Phi_{sum} - \Psi_{sum} & otherwise. \end{cases} \tag{2.10}$$

Then the SAD is acquired through

$$SAD = \begin{cases} \text{Diff}_{sum} + \sum\limits_{\varphi_i < \psi_i} 2(\psi_i - \varphi_i) & if\ \Psi_{sum} > \Phi_{sum} \\ \text{Diff}_{sum} + \sum\limits_{\varphi_i > \psi_i} 2(\psi_i - \varphi_i) & if\ \Psi_{sum} < \Phi_{sum} \\ \sum\limits_{i=0}^{N-1} |\psi_i - \varphi_i| & if\ \Psi_{sum} = \Phi_{sum} \end{cases} \tag{2.11}$$

From Equation 2.11, it is clear that based on the size relationship of the difference of sum, only the elements that have a opposite size relationship between two sets are involved to calculate the SAD. Moreover, this part of elements is the minority of the whole two sets.

For the sake of having a more quantized understanding of the "minority", statistics on percentage of pixels per block in video signals accounted as minority has been collected [4]. The test is performed on six video sequences (Figure 2.2) frame by frame to observe the percentage of pixels per MB (16×16 applied) that can be accounted as "minority" to the SAD calculation. The experimental results shown in Figure 2.3 indicate that the "minority" is referred to 10%∼32% pixels in these video sequences. It means when calculating the SAD only 10%∼32% pixels are involved in Equation 2.11, hence the amount of computation of SADs can be reduced by a large scale.



(a) akiyo (b) flower (c) cricket

(d) mobile (e) suzie (f) foreman

Figure 2.2: 6 video sequences for test on statistics of "minority".

2.2.2 The New SAD architecture to accelerate the FS ME algorithm

Section 1.3 has given an introduction of Full Search algorithm for details. It is a ME search algorithm which can guarantee the best video quality and lowest bitrate but suffer from high complexity and poor encoding speed. Many fast ME search algorithms has been proposed in the literature [24, 27–43], for instance three-step search, diamond search, hexagonal patterns Search and so on (introduced in Section 1.3). They speed up the ME search process through reducing the number of search points and win a reduction on the encoder complexity and an improvement on the throughput at a cost of video quality

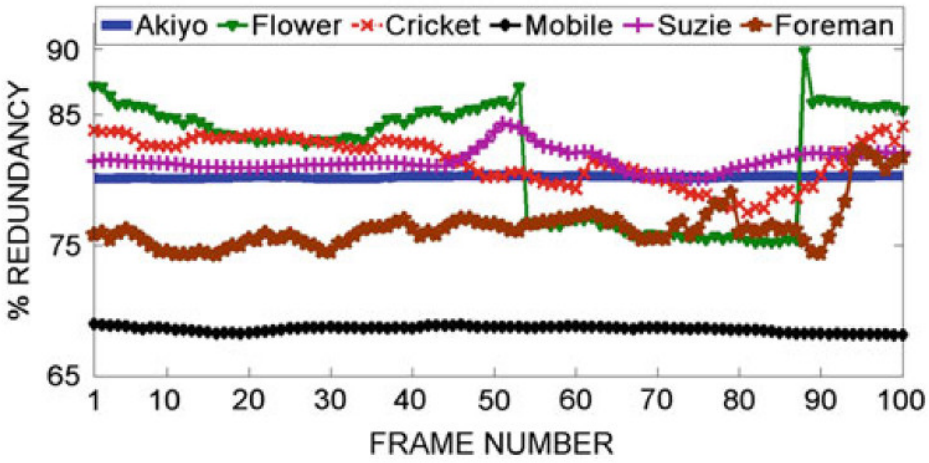


Figure 2.3: The percentage of pixel per MB that can be accounted as "minority" to the SAD calculation in six video sequences frame by frame [4].

loss and bitrate increase. In this section a new SAD architecture to accelerate the FS algorithm is proposed based on the strategy described in Section 2.2.1 by removing the computing redundancies in the conventional SAD architecture. The block diagram of the new SAD algorithm for ME is illustrated in Figure 2.4.

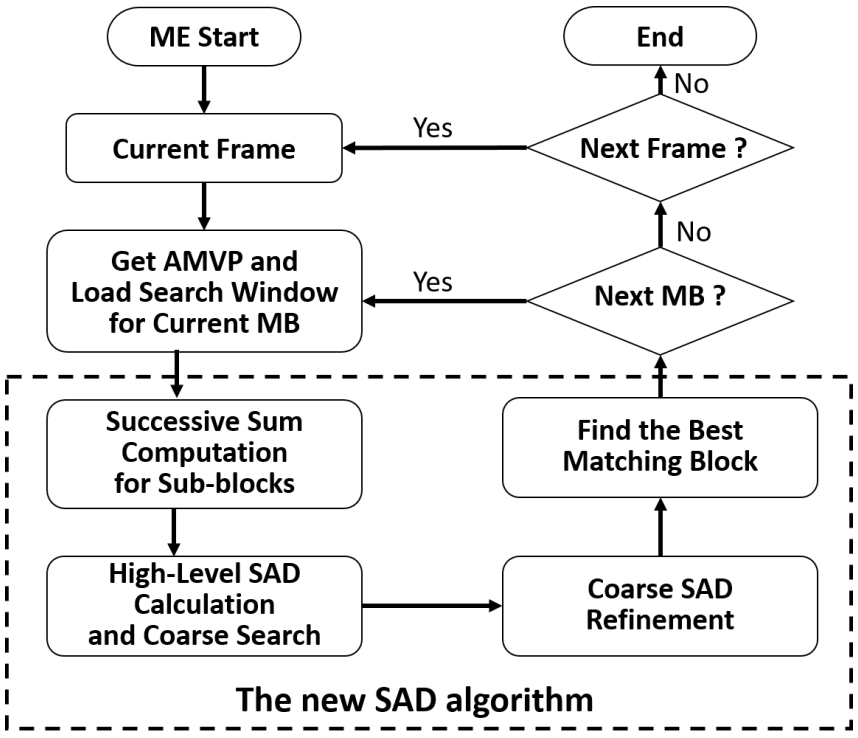


Figure 2.4: Block diagram of the new SAD algorithm.

Fast Successive Sum Computation

As for the FS algorithm, it is necessary to search every candidate block within the search window one by one. In the new SAD algorithm the sum of the block needs to be known first. Since between two consecutive candidate block most of the pixels are overlapped, performing sum operation separately for each blocks must bring about high redundancy of additions. Thus a fast successive sum computational strategy is explored to alleviate computational load, which is exhibited in Figure 2.5. One thing has to be pointed out

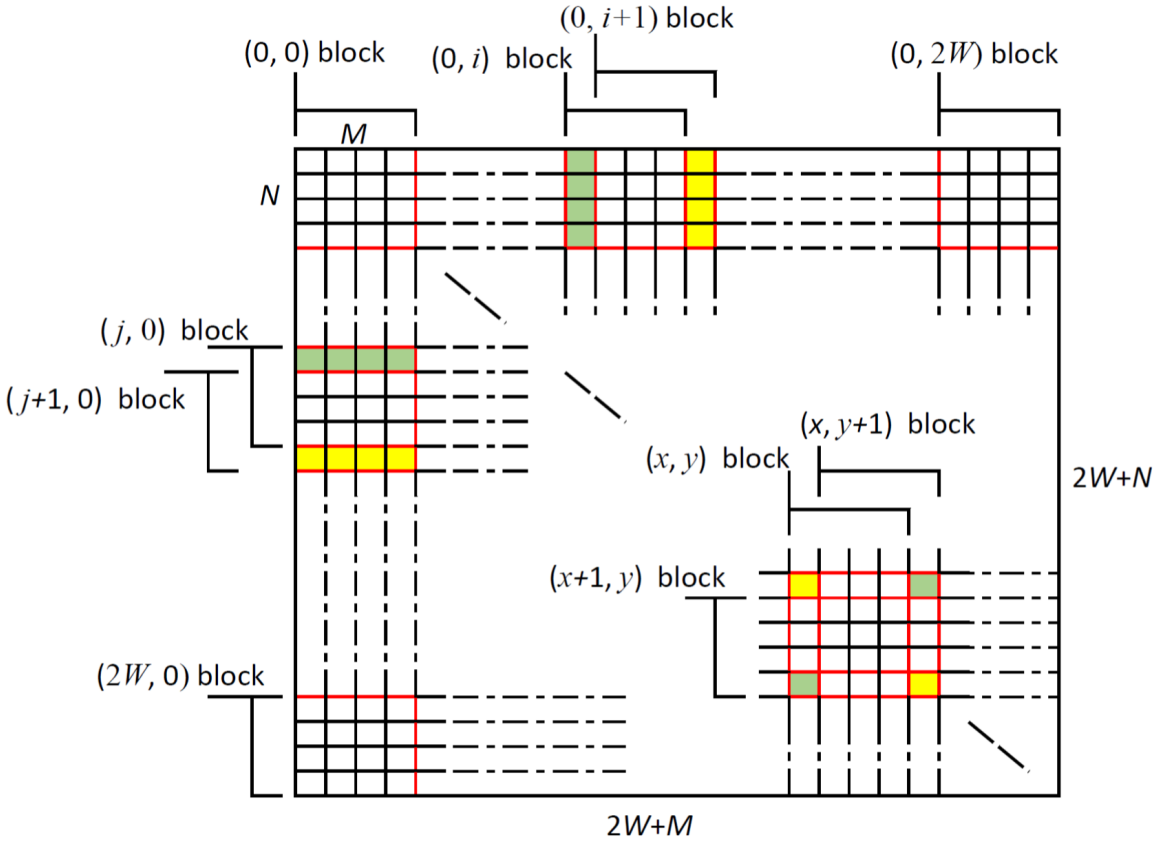


Figure 2.5: Successive SAD calculation.

that, according to Section 1.2 HEVC standard support flexible partitions from 4×4 up to 64×64 and also including the symmetrical block size such as 8×4, 16×8, 32×16 and 64×32 and their transformed block sizes. Therefore for a MB size $N \times M$ with a search range of $\pm W$, the search window's dimension is $(2W + N)\times(2W + M)$. In order to find the best matching block, the total number of candidate block is $(2W + 1)^2$. As shown in Figure 2.5, when fast successive sum computation starts the first (at (0, 0) position) candidate block's sum is calculated first with a cost of (N×M-1) additions. In the following for the candidate blocks in the first row, the sum of the latter block can be obtained by using the sum of the former block plus the pixels of the last column of current block (highlighted by

the yellow strip) and minus the pixels of the first column of the former block(highlighted by the green strip) as shown in Figure 2.5. For instance let $Sum_{(0,i)}$ and $Sum_{(0,i+1)}$ denote the sum of the $(0, i)$ and $(0, i + 1)$ block respectively, $Sum_{(0,i+1)}$ is calculated using

$$Sum_{(0,i+1)} = Sum_{(0,i)} + \sum_{s=0}^{N-1} \mathrm{SW}(s, i + \mathrm{M}) - \sum_{s=0}^{N-1} \mathrm{SW}(s, i), \tag{2.12}$$

where $\mathrm{SW}(s, i)$ is the pixel intensity at point (s, i). Correspondingly for the candidate blocks in the first column, its sum is acquired by

$$Sum_{(j+1,0)} = Sum_{(j,0)} + \sum_{s=0}^{M-1} \mathrm{SW}(j + \mathrm{N}, s) - \sum_{s=0}^{M-1} \mathrm{SW}(j, s). \tag{2.13}$$

As a result, the number of operations to compute the sum of candidate blocks in the first row and column except the (0, 0) block is 2(N+1) and 2(M+1) additions (or subtractions) respectively. Then as well for the rest candidate blocks, the sum is obtained based on the sum of top, left and top-left neighboring blocks. As shown in Figure 2.5, the sum of block $(x + 1, y + 1)$ is given by

$$\begin{aligned} Sum_{(x+1,y+1)} = \; & Sum_{(x,y+1)} + Sum_{(x+1,y)} - Sum_{(x,y)} \\ & + \mathrm{SW}(x, y) - \mathrm{SW}(x + M, y) - \mathrm{SW}(x, y + N) + \mathrm{SW}(x + M, y + N). \end{aligned} \tag{2.14}$$

Apparently Equation 2.14 only spend 6 additions (or subtractions) on computing sum of each candidate block not belonging to first row or column. Therefore the total number of operations to calculate the sum of all candidate blocks within the search window using fast successive sum strategy, denoted as Γ_{fast}, is

$$\Gamma_{\mathrm{fast}} = (N \times M - 1) + 2W \times [2(N + 1) + 2(M + 1)] + (2W)^2 \times 6. \tag{2.15}$$

Meanwhile the total number of operations to calculate the sum of every candidate block separately in the original FS algorithm turns out to be

$$\Gamma_{\mathrm{org}} = (2W + 1)^2 \times (N \times M - 1). \tag{2.16}$$

Taking a real case as an example, in HEVC standard the CTU's size is 64×64 and search range is typical of ±64, hence

$$\begin{aligned} \Gamma_{\mathrm{fast}} \; & = (64 \times 64 - 1) + 2 \times 64 \times [2 \times (64 + 1) + 2 \times (64 + 1)] + (2 \times 64)^2 \times 6 \\ & = 135679 \end{aligned} \tag{2.17}$$

and

$$\Gamma_{\text{org}} = (2 \times 64 + 1)^2 \times (64 \times 64 - 1) = 68144895. \tag{2.18}$$

Furthermore it is easy to find that the computational savings is $\frac{68144895 - 135679}{68144895} \times 100\% = 99.8\%$.

High Level SAD Calculation and Coarse Search

In the new SAD algorithm, for the purpose of further improving the search efficiency a coarse search will be applied initially. Since most of the ME modules in the video encoding system support symmetric block size, Equation 2.3 for a MB of size N×M has to be rewrote as

$$SAD(MV_x, MV_y) = \sum_{i=u}^{u+N-1} \sum_{i=v}^{v+M-1} [f(i,j,t) - f(i + MV_x, j + MV_y, t-1)]. \tag{2.19}$$

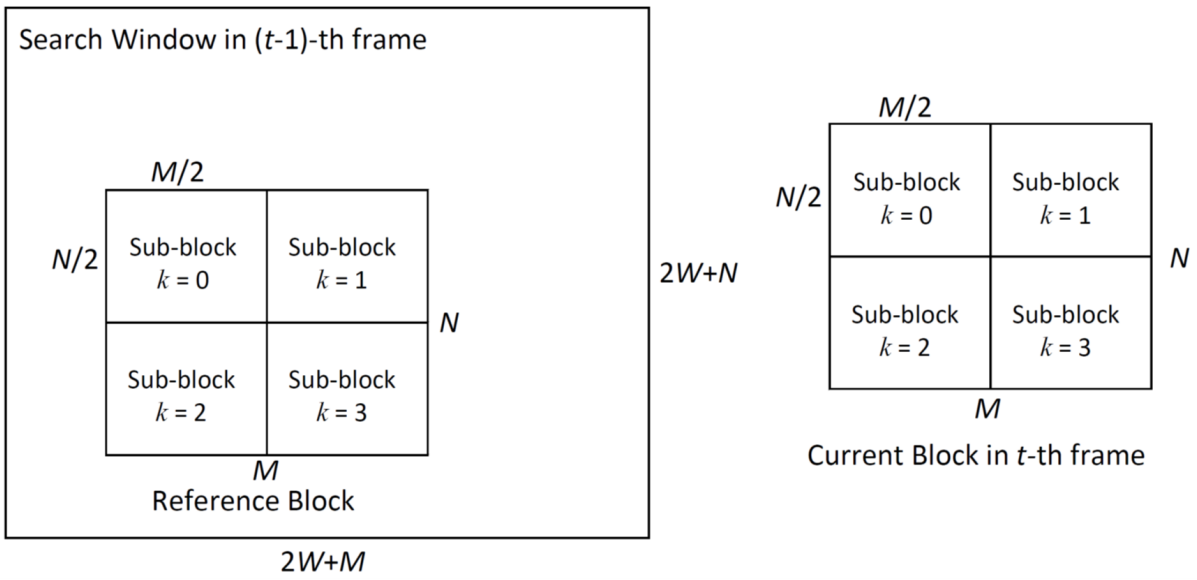


Figure 2.6: An example for the sub-blocks.

In order to perform the coarse search, both of the current MB and reference MB are firstly divided into 4 sub-blocks respectively thus each of $(N/2) \times (M/2)$, denoted as $f_k(i,j,t)$ and $f_k(i + MV_x, j + MV_y, t-1)$ $(k \in [0,3])$, which is illustrated in Figure 2.6. Then as indicated in Figure 2.4 the sum of each sub-block of the current MB and all the reference MB inside the search window is obtained through

$$Sum(i,j,t)_k = \sum_{i=0}^{N/2-1} \sum_{j=0}^{M/2-1} f(i,j,t)_k \tag{2.20}$$

and

$$Sum(i+MV_x, j+MV_y, t-1)_k = \sum_{i=0}^{N/2-1} \sum_{j=0}^{M/2-1} f(i+MV_x, j+MV_y, t-1)_k \qquad (2.21)$$

respectively, where $Sum(i+MV_x, j+MV_y, t-1)_k$'s computation resorts to the fast successive sum module. In the next the high level absolute difference between the two k-th current and reference sub-blocks, denoted as $CSAD(i+MV_x, j+MV_y)_k$ (Coarse Sum of Absolute Difference), can be calculated by

$$CSAD(i+MV_x, j+MV_y)_k = |Sum(i,j,t)_k - Sum(i+MV_x, j+MV_y, t-1)_k|. \quad (2.22)$$

Consequently, the coarse sum of absolute difference between the current block and reference block is obtained using

$$CSAD(i+MV_x, j+MV_y) = \sum_{k=0}^{3} CSAD(i+MV_x, j+MV_y)_k. \qquad (2.23)$$

where (MV_x, MV_y) is a displacement within the search window, thus $(-W, -W) \leq (MV_x, MV_y) \leq (+W, +W)$.

Let's assign the SAD result at the start point, namely AMVP, to SAD_{min} as its initial value, the coarse search is conducted in the way that if $CSAD(i+MV_x, j+MV_y) < SAD_{min}$ the current motion vector (MV_x, MV_y) is added to the MV set, Ω, and the SAD_{min} is updated to the newest minimum value, or current displacement is discarded. After coarse search is finished, the MVs stored in the Ω will be used in the final best matching block search. The block diagram of coarse search is shown in Figure 2.7. This process helps to exclude a number of impossible candidates which can alleviate the workload of refining search in the next stage and save the computational cost.

Coarse SAD Refinement

When finishing coarse search, it is ready to carry out the refinement of the coarse SAD using the positions with MV$\in \Omega$. Grounded on the introduction in Section 2.2.1, the refinement can be fulfilled basing on the size relationship between $Sum(i,j,t)_k$ and $Sum(i+MV_x, j+MV_y, t-1)_k$ and the "minority" pixels which have a opposite size relationship to their block size relationship. So the SAD between the two k-th corresponding current and reference sub-blocks, being $SAD_k(i+MV_x, j+MV_y)$ can be expressed as:

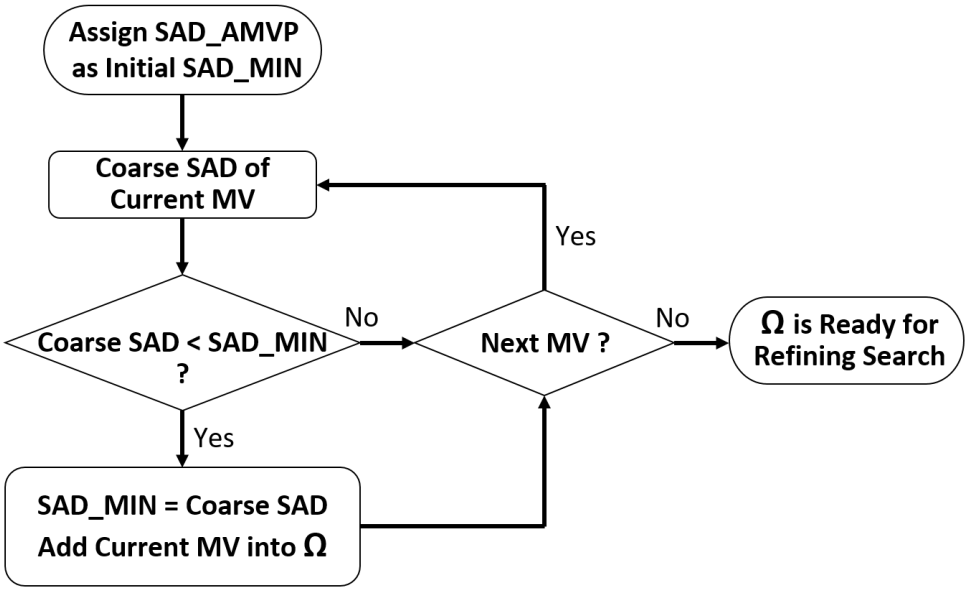


Figure 2.7: Block diagram of coarse search.

If $Sum(i,j,t)_k > Sum(i+MV_x, j+MV_y, t-1)_k$:

$$\begin{aligned} SAD(i+MV_x, j+MV_y)_k &= CSAD(i+MV_x, j+MV_y)_k \\ &+ \sum_{\Lambda} 2 \times [f(i+MV_x, j+MV_y, t-1) - f(i,j,t)] \end{aligned} \tag{2.24}$$

where Λ refers to a set containing all pixels that $f(i,j,t) < f(i+MV_x, j+MV_y, t-1)$ within the two k-th sub-blocks.

If $Sum(i,j,t)_k < Sum(i+MV_x, j+MV_y, t-1)_k$:

$$\begin{aligned} SAD(i+MV_x, j+MV_y)_k &= CSAD(i+MV_x, j+MV_y)_k \\ &+ \sum_{\Lambda'} 2 \times [f(i,j,t) - f(i+MV_x, j+MV_y, t-1)] \end{aligned} \tag{2.25}$$

where Λ' refers to a set containing all pixels that $f(i,j,t) > f(i+MV_x, j+MV_y, t-1)$ within the two k-th sub-blocks.

If $Sum(i,j,t)_k = Sum(i+MV_x, j+MV_y, t-1)_k$:

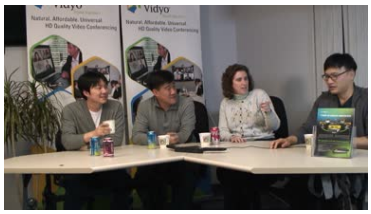
$$SAD(i+MV_x, j+MV_y)_k = \sum_{i=u}^{u+N/2-1} \sum_{i=v}^{v+M/2-1} |f(i,j,t) - f(i+MV_x, j+MV_y, t-1)|. \tag{2.26}$$

And finally the distortion between the current and reference block, denoted as $SAD(i+$

$MV_x, j + MV_y$), is obtained by

$$SAD(i + MV_x, j + MV_y) = \sum_{k=0}^{3} SAD(i + MV_x, j + MV_y)_k. \tag{2.27}$$

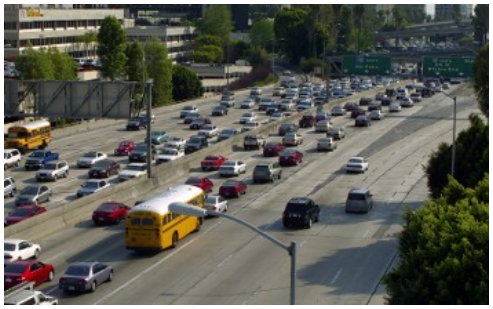
From Equation 2.24 and 2.25, it can observed that the SAD refinement only involves the minority pixels (discussed in Section 2.2.1) which is just a small proportion of the total amount of pixels in a MB. Therefore, it can be expected that a large scale of the computing load in ME process will be alleviated.



(a) Four People, 1280x720, 60fps, Class E

(b) Basketball Drive, 1920x1080, 50fps, Class B

(c) Park Scene, 1920x1080, 24fps, Class B



(d) Traffic, 2560x1600, 30fps, Class A

(e) Ready steady go, 3840x2160, 120fps, 4K Test Sequence

Figure 2.8: HD video sequences for test used in the simulation.

2.2.3 Rate-Distortion Performance Evaluation with HM Test Model

The proposed new SAD computing algorithm for Full Search ME algorithm are implemented, verified and tested in HEVC Test Model (HM16.2). The simulations are performed under JCT-VC common conditions (Low Delay using P pictures with uni-prediction) on the platform of Ubuntu OS with Intel Core i5 2.67GHz CPU and 8 Gigabyte main memory. In order to evaluate its performance, several high definition standard video sequences for test from different classes are encoded using a prediction structure of Low Delay P picture (LDP) defined by Common Test Conditions (CTC). The LDP specifies

Four People 1280×720 @60Hz							
QP	Algorithm	BitRate		YUV-PSNR		Encoding Time	
		kbps	Δ(%)	dB	Δ(%)	seconds	Δ(%)
37	Full search	862.7040	–	36.4348	–	4454.33	–
	Proposed	864.5760	0.22%	36.4293	-0.02%	1509.19	-66.12%
32	Full search	1454.9280	–	39.1425	–	4422.22	–
	Proposed	1458.6720	0.26%	39.1439	0.00%	1342.92	-69.63%
27	Full search	2463.4080	–	41.5813	–	4471.51	–
	Proposed	2468.9760	0.23%	41.5761	-0.01%	1221.93	-72.67%
22	Full search	4793.1360	–	43.5971	–	4478.89	–
	Proposed	4795.6800	0.05%	43.5941	-0.01%	1172.37	-73.82%
Average	Proposed	–	0.19%	–	-0.01%	–	-70.56%

Table 2.1: Simulation results using different QP parameters for test sequence of Four People.

Basketball Drive 1920×1080@50Hz							
QP	Algorithm	BitRate		YUV-PSNR		Encoding Time	
		kbps	Δ(%)	dB	Δ(%)	seconds	Δ(%)
37	Full search	1759.0400	–	36.1091	–	10061.71	–
	Proposed	1789.5200	1.73%	36.0666	-0.12%	5623.51	-44.11%
32	Full search	3248.8400	–	37.9569	–	10015.84	–
	Proposed	3297.2000	1.49%	37.9238	-0.09%	5106.57	-49.02%
27	Full search	6524.1200	–	39.5933	–	10145.28	–
	Proposed	6606.0800	1.26%	39.5816	-0.03%	4627.32	-54.39%
22	Full search	17836.1200	–	41.0429	–	10249.15	–
	Proposed	17953.4400	0.66%	41.0363	-0.02%	4401.93	-57.05%
Average	Proposed	–	1.28%	–	-0.06%	–	-51.14%

Table 2.2: Simulation results using different QP parameters for test sequence of Basketball Drive.

that video frames are configured to one I picture followed by 8 P pictures and reordering of pictures is not allowed and only past pictures are used for prediction [22,53–55]. Figure 2.8 lists all video sequences used in the test: (1) Four People(1280x720@60Hz, Class E), (2) Basketball Drive (1920x1080@50Hz, Class B), (3) Park Scene (1920x1080@24Hz, Class B), (4) Traffic (2560x1600@30Hz, Class A) and (5) Ready steady go (3840x2160@120Hz, 4K Test Sequence). The simulation run with 10 frames of each of the test sequences using 4 different Quantized Parameters (QP, 22, 27, 32 and 37)and a search range of ±64. In the test, data of BitRate, YUV-PSNR [52] and total encoding time are collected for performance comparison. The results , presented in Table 2.1-2.5, 2.2 clearly show

Park Scene 1920×1080@24Hz							
QP	Algorithm	BitRate		YUV-PSNR		Encoding Time	
		kbps	△(%)	dB	△(%)	seconds	△(%)
37	Full search	1160.9664	–	33.4954	–	10053.54	–
	Proposed	1164.2112	0.28%	33.4719	-0.07%	6185.51	-38.47%
32	Full search	2500.8768	–	35.7674	–	10001.12	–
	Proposed	2507.0208	0.25%	35.7466	-0.06%	5674.58	-43.26%
27	Full search	5358.2592	–	38.1899	–	10138.20	–
	Proposed	5383.8528	0.48%	38.1773	-0.03%	5244.51	-48.27%
22	Full search	12178.6368	–	40.6841	–	10268.81	–
	Proposed	12237.6768	0.48%	40.6765	-0.02%	5062.84	-50.70%
Average	Proposed	–	0.37%	–	-0.04%	–	-45.18%

Table 2.3: Simulation results using different QP parameters for test sequence of Park Scene.

Traffic 2560×1600 @30Hz							
QP	Algorithm	BitRate		YUV-PSNR		Encoding Time	
		kbps	△(%)	dB	△(%)	seconds	△(%)
37	Full search	2679.7680	–	34.5202	–	20124.47	–
	Proposed	2695.4880	0.59%	34.5096	-0.03%	7866.48	-60.91%
32	Full search	5021.1360	–	36.9135	–	20009.43	–
	Proposed	5054.4960	0.66%	36.8998	-0.04%	6907.14	-65.48%
27	Full search	10146.9600	–	39.2883	–	20249.14	–
	Proposed	10203.9600	0.56%	39.2756	-0.03%	6273.78	-69.02%
22	Full search	23945.0400	–	41.8368	–	20433.65	–
	Proposed	24091.3440	0.61%	41.8290	-0.02%	5964.33	-70.81%
Average	Proposed	–	0.61%	–	-0.03%	–	-66.55%

Table 2.4: Simulation results using different QP parameters for test sequence of Traffic.

that compared to the ME with original FS algorithm, the proposed one can accomplish a reduction on the total encoding time on average by at least 45.18% or 70.56% in the best case, with only a minor increase on bit-rate and a negligible PSNR loss. Furthermore there is a more clear illustration on the Rate-Distortion performance comparison exhibited in Figure 2.9, where the curves of PSNR v.s. bit-rate for all test sequences are almost totally coincided. It means the proposed new SAD algorithm achieves a significant reduction on computational load but with a negligible performance loss.

Ready Steady Go 3840×2160 @120Hz							
QP	Algorithm	BitRate		YUV-PSNR		Encoding Time	
		kbps	△(%)	dB	△(%)	seconds	△(%)
37	Full search	12231.7440	–	36.6316	–	41151.41	–
	Proposed	12291.0720	0.49%	36.6056	-0.07%	19285.25	-53.14%
32	Full search	22159.2960	–	38.6549	–	40398.51	–
	Proposed	22270.5600	0.50%	38.6410	-0.04%	17737.61	-56.09%
27	Full search	43401.3120	–	40.3937	–	40853.98	–
	Proposed	43619.5200	0.50%	40.3836	-0.03%	16576.12	-59.43%
22	Full search	114894.3360	–	41.9054	–	41265.31	–
	Proposed	115226.7840	0.29%	41.8967	-0.02%	16357.89	-60.36%
Average	Proposed	–	0.44%	–	-0.04%	–	-57.25%

Table 2.5: Simulation results using different QP parameters for test sequence of Ready Steady Go.

2.3 Hardware Implementation

As discussed in the last section, the new SAD algorithm consists of two parts: high-level SAD coarse search and coarse SAD refinement and final search. The high-level SAD computation utilizes the fast successive sum strategy. And the sum results of the sub-blocks obtained in the high-level SAD stage can be reused for the size relationship between sub-blocks in the SAD refinement. The hardware implementation of whole ME module is depicted in Figure 2.10.

ME Engine Control Unit

The ME Engine control is mainly responsible for the control signals generation for all the other function units. Initially after the power-on reset signal switches on the encoding system, it firstly transmits a enable signal and a corresponding address to Memory Read&Write Unit to load the first frame of pixels for reference and the first LCU (64×64) for encoding. In the next it needs to arrange the coding loop order for all the CUs of current CTU which is normally from CU64×64, CU64×32, CU32×64... till to CU16×8, CU8×8. Then after the AMVP unit determines the search start point, the important task of ME Engine control is to configure the search window and guarantee its border lies inside the reference frame according to the AMVP's coordinate and search range, and then load the pixels of search window for current CU from memory bank to the on-chip buffer. In the end, when the motion estimation of current CTU is finished it will guide the MV Read&Write Control Unit to store the final optimal MVs.

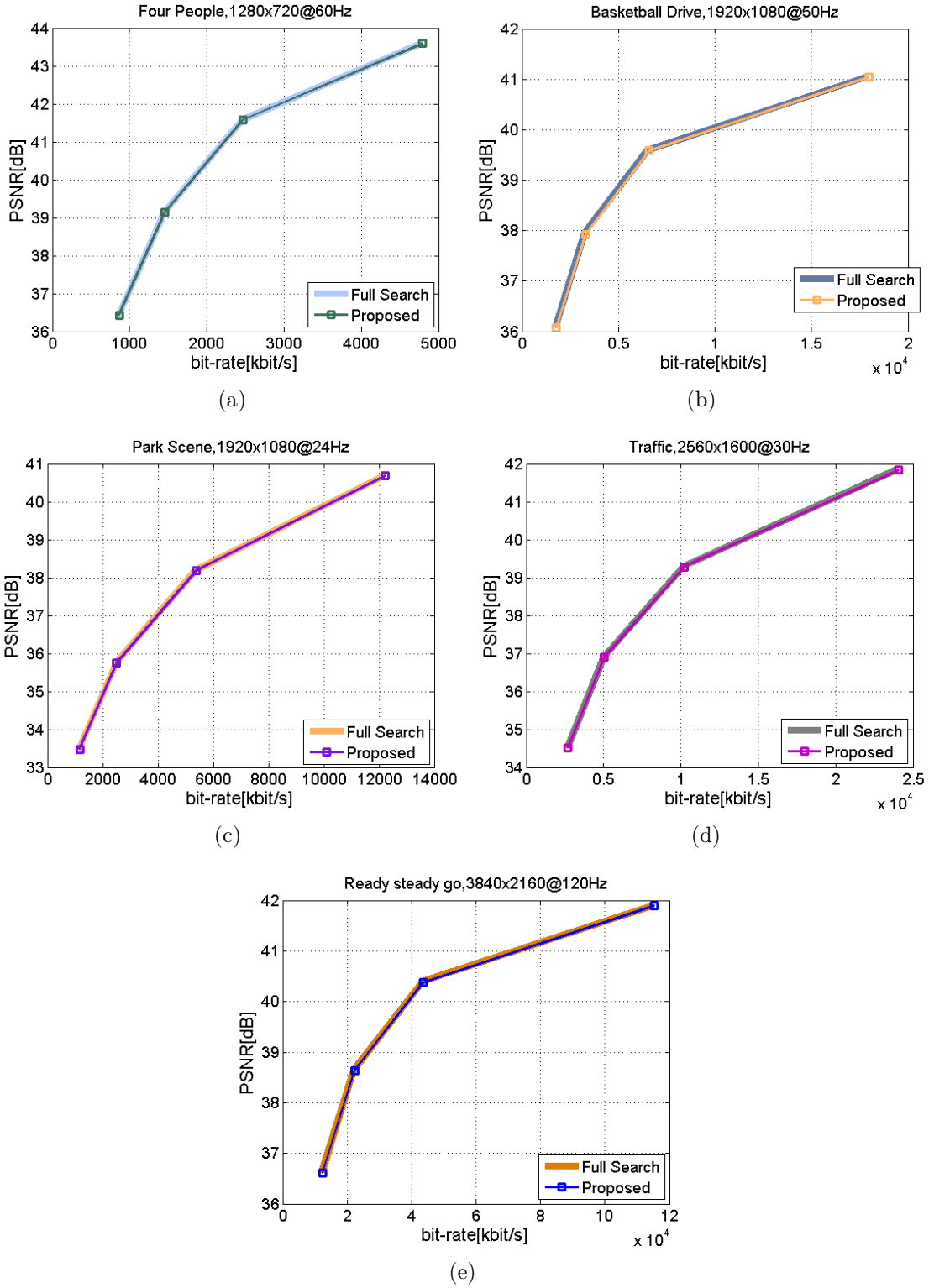


Figure 2.9: Rate-Distortion curves for all test sequences.

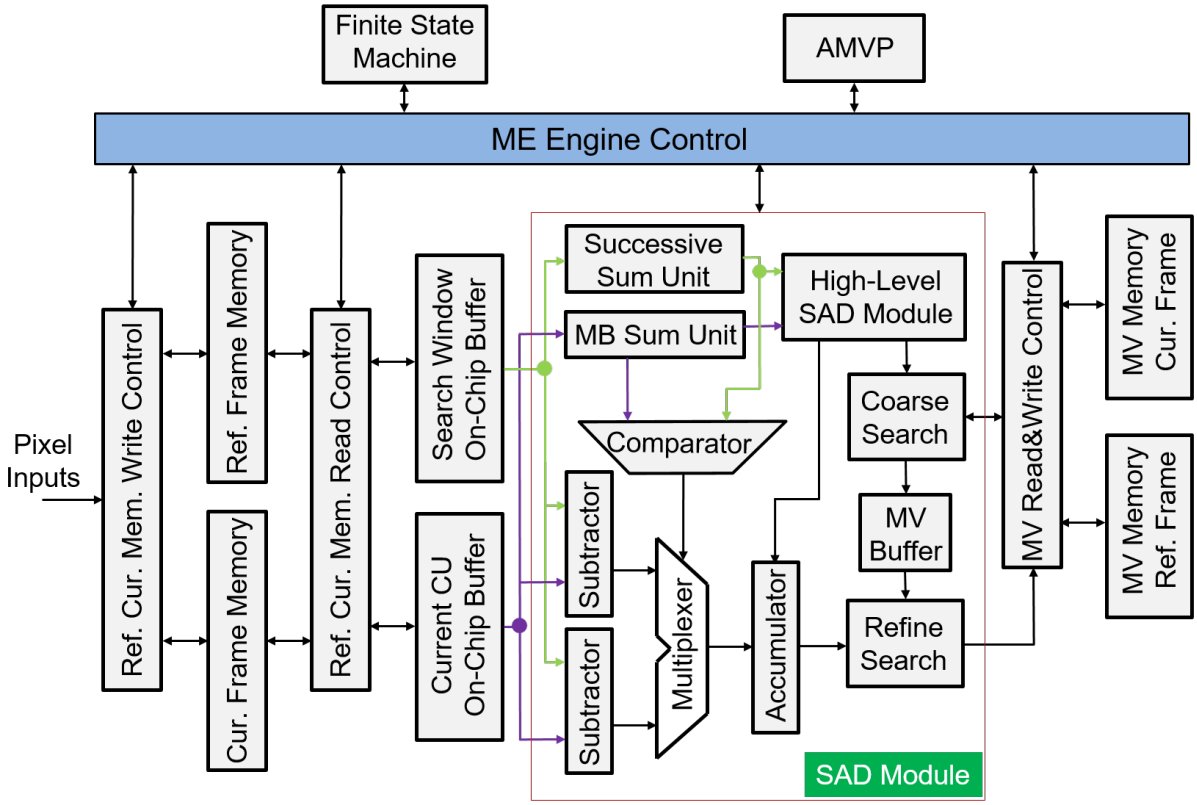


Figure 2.10: Hardware architecture of proposed SAD algorithm.

Finite State Machine

For the sake of managing all the functional units to work in orderly and efficiently, the Finite State Machine is to control the state switching of the whole system between memory reading and writing, matching block searching, CU encoding looping and CU mode decision. The ME Engine Control Unit arranges different enable signals for different function unit according to the current state.

AMVP

When the encoding process for a new CU starts, the AMVP unit will check and read the available spatial and temporal motion vector predictors from MV memory and determine the best MV as the start search point. While the search window is decided using the AMVP by the ME Engine Control unit.

Memory Read Write Control

Here in the proposed architecture, either the current and reference frame memory or MV memory for current and reference frame are implemented in a ping-pong mode to save

the off-chip memory size and improve the memory reading and writing efficiency. Hence there are two separate memory banks, denoting as Bank#0 and Bank#1. When encoding the current frame if Bank#0 stores pixels for the past reference frame and Bank#1 holds the data from current frame, then when encoding the next frame the pixels stored in the Bank#1 will be used as the reference and meantime pixels of the new frame to be encoded will be loaded into Bank#0, and so on and so forth. Meanwhile as for the MV memory for current and reference frame the same mode is also applied. The functionality of the this working mechanism is fulfilled by the Memory Read Write Control unit.

SAD Module

As discussed in this chapter, the SAD module is the key unit in the top level architecture of ME structure. After the search window for current PU is finished loading, the MB Sum unit and Successive Sum unit will perform summing operation for the sub-blocks of current MB and all the candidate blocks respectively, meantime the coarse SAD between sub-blocks of current MB and reference MB is also generated. Based on these coarse SADs, impossible candidates are excluded by the coarse search process and motion vectors of valid candidates are pushed into in the MV buffers for the refining search. With the sum results of sub-blocks, a comparator generates a select signal for the multiplexer which together with two subtractors are responsible for the coarse SAD refinement by adding up the high level SAD and difference between corresponding pixels produced from Multiplexer. At last the Refine Search module will generate the best MV for the current PU from candidates stored in MV buffer based on the refined SADs.

2.4 Summary

A new SAD computing architecture is developed to alleviate the computational load of full search ME algorithm. Simulating in the HM test model using several standard FHD or UHD test sequences, the proposed architecture can reduce up to 70% of the computations with respect to the original FS algorithm with an imperceptible bit-rate increase and PSNR loss. This architecture is applicable to the video encoding system which pursuits highest video quality.

Chapter 3

An Enhanced TZ (Test Zone) Search Algorithm for Fast Motion Estimation of HEVC

Generally speaking, in the block based video coding system there are two categories of ME algorithms: Full Search and Fast Search [56–58]. The former one searches all the possible candidates within the pre-determined search window, which can guarantee a highest output video quality and a smallest amount of bit-rate. But its complexity usually is very high and the scale of computations is huge. On the other hand, the fast search algorithms, such as Three-Step Search, Diamond Search and so on (discussed in Section 1.3), reduce the complexity and computations by partially searching of candidates in the search window. Consequently, its throughput is incremented but with a cost of video quality loss and bit-rate increase. Along with the prevalence of high definition videos, full search algorithm is rarely utilized in encoding applications because of extremely high complexity. Whereas fast search algorithms are widely used. Since currently some of the best fast search algorithms can reach a comparable good video quality as full search only with a slight bit-rate increase, and still maintain a low complexity. And TZ Search algorithm is one of them.

3.1 Introduction of TZ Search Algorithm

TZ Search is a novel widely-used fast search algorithm, which is adopted by the reference software of the newest video coding standard HEVC because of its good performance [40, 59–64]. It is indeed a hybrid search algorithm which combines the diamond/square search and raster search. The whole procedure of TZ Search consists of several steps and its flowchart is shown in Figure 3.1. From the flowchart, we can see TZ Search firstly set

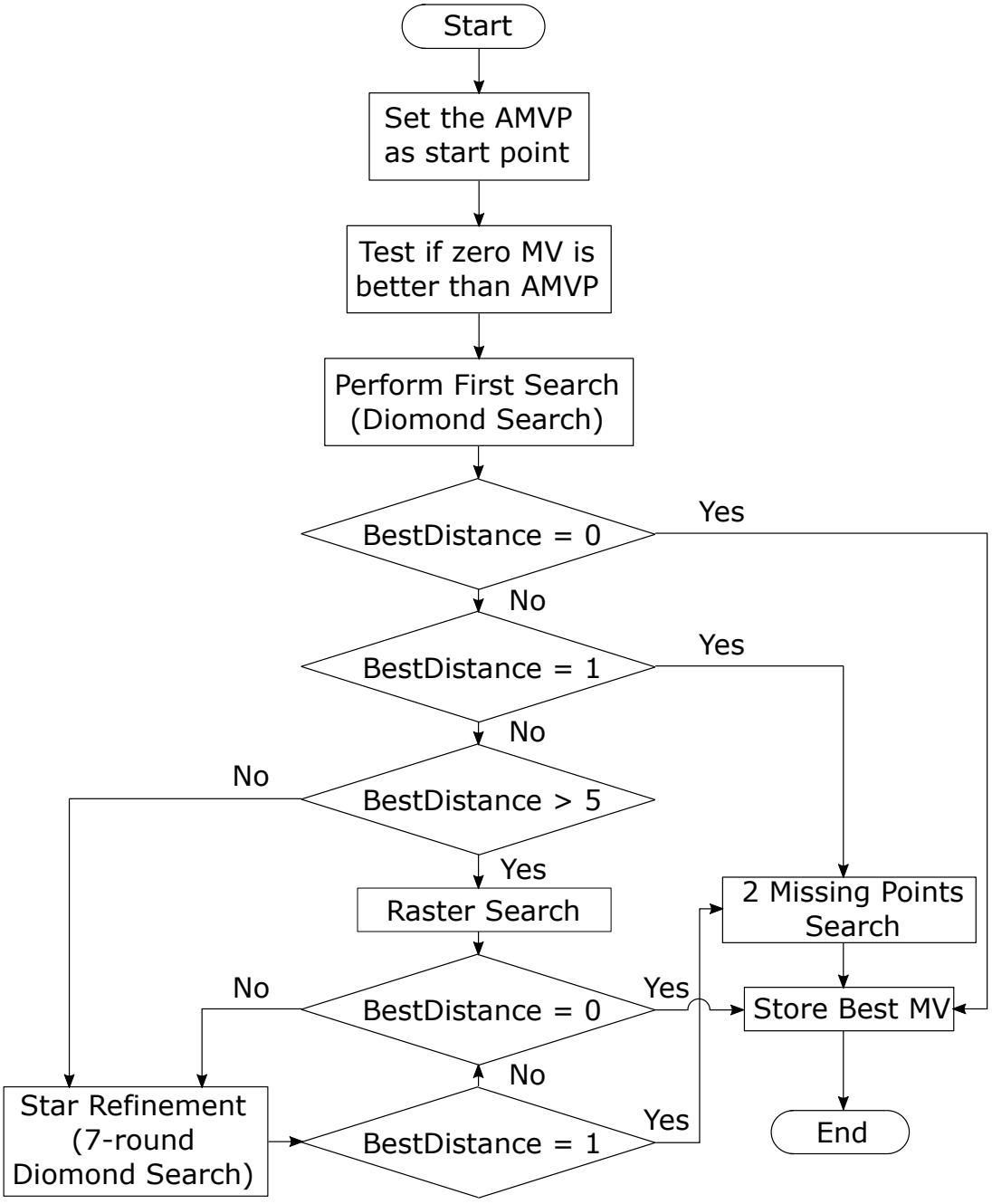


Figure 3.1: The flowchart diagram of TZ Search.

the start point as AMVP and check whether the zero motion vector is superior than the AMVP to be a even better start point. Then with the best start point and search range, the search process is ready to launch.

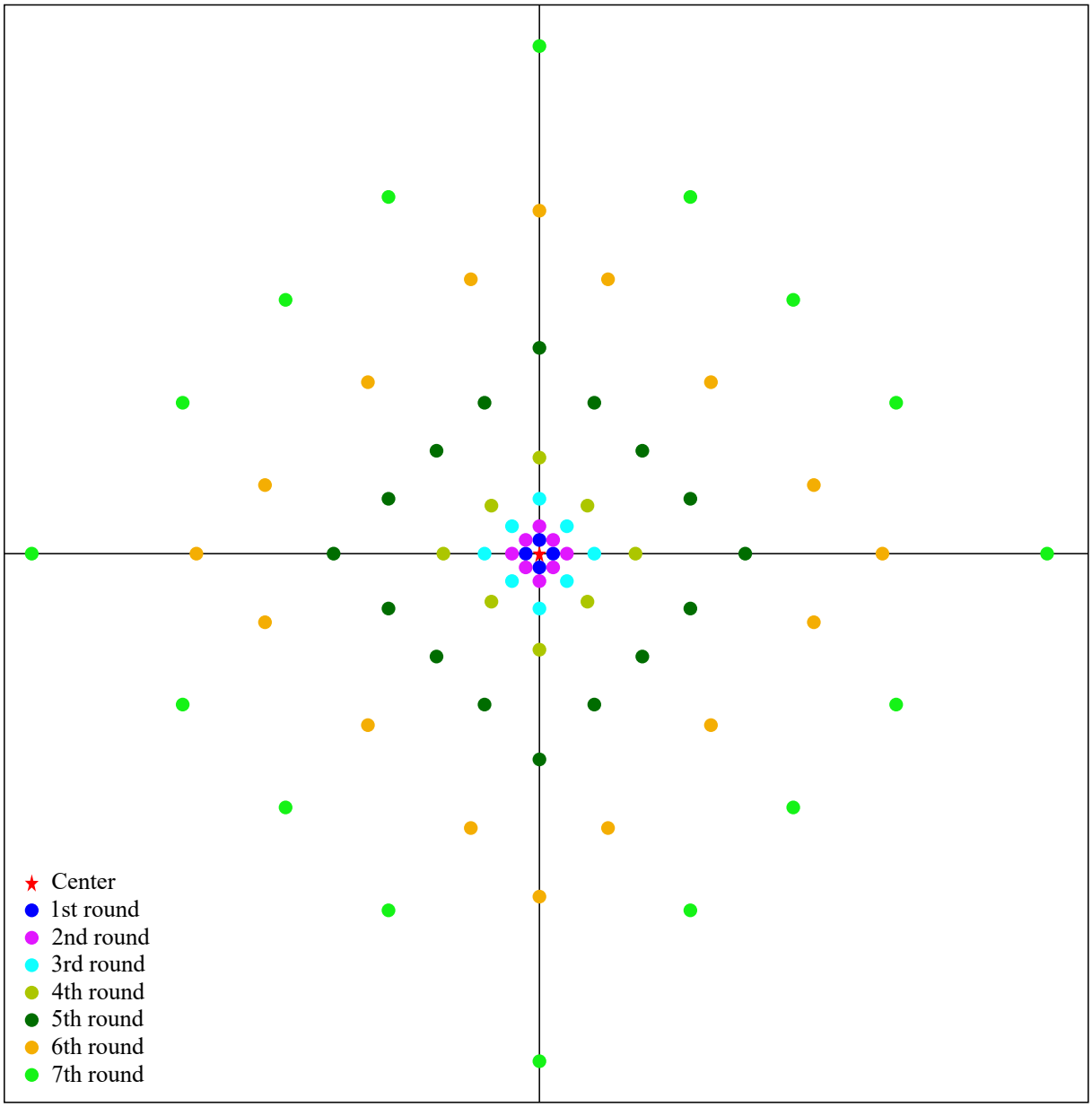


Figure 3.2: The 7-round Diamond Search in the TZ Search algorithm.

First Search

The initial search process of TZ Search is to perform a First Search. In the HEVC reference software there are two similar choices of search algorithm provided for First Search: Square Search and Diamond Search, but the latter one is set as default. Figure 3.2 illustrates the search patterns and strategy of 7-round Diamond Search, where the red star in the center symbolizes the start point, namely the search center, and the other colorful dots mark the candidate positions with different color suggesting its corresponding search round. One characteristic of Diamond Search algorithm is that the number of rounds of search is not

limited which is normally determined by the search range or the size of search window. Let the number of rounds and search range be $\mathcal{N}$ and $\pm SR$ respectively, the relationship between them is given by

$$\mathcal{N} = \lfloor \log_2(SR) \rfloor + 1. \tag{3.1}$$

And the distance between candidate and the search center for each round, denoted as $\mathcal{D}$, is incremented exponentially and determined by

$$\mathcal{D} = 2^{r-1} \quad r \in [1,\ \mathcal{N}]. \tag{3.2}$$

For instance, typically the default search range used in HEVC test model is ± 64, thus $\mathcal{N}$=7. It is indicated in Figure 3.2 that the 1st round of Diamond Search has 4 search patterns around the start point with a distance equal to 1, whose positions are the top, bottom, right and left. From 2nd to 4th round, in addition to the 1st round the number of search patterns for each round is increased to 8 with 4 half-positions (half topleft, half topright, half bottomleft, half bottomright), and the distance of each round is 2, 4 and 8 respectively. Similarly from 5th to 7th round, 8 more quarter positions are further introduced for each round in addition to the positions used in 2nd to 4th round, and the distance of each round is 16, 32 and 64 separately. Consequently, it is easy to figure out that the total number of search patterns in the First Search is 76 for a search window with full search range. When the search for all available patterns is finished, there are three possible branches based on the searching result of First Search which are listed in the following.

(1) **Termination**: If the best matching block is with a distance of 0, which means the MV predictor is accurate enough that no candidates block has a better matching than the start points, the whole TZ Search process is immediately terminated with the start point as the final best MV;

(2) **2 Missing Points Search**: If the distance of the best matching block is equal to 1, the next step of First Search is to carry out a 2 Missing Points search, which acts as a final refinement on its neighbor positions before the completion of TZ Search;

(3) **Raster Search**: In case that the best matching block obtained from the First Search is more than 5 pixels away from the start point, a Raster Search within the whole search window is required to operate. Since a best distance of more than 5 signifies that the accuracy of MV prediction is not good, taking global search as a remedy is necessary for finding the best matching block;

(4) **Refinement Search**: Other than all the aforementioned cases, if the distance falls into the scope of 2 to 5 pixels, instead of the refinement on its neighbors, a refinement

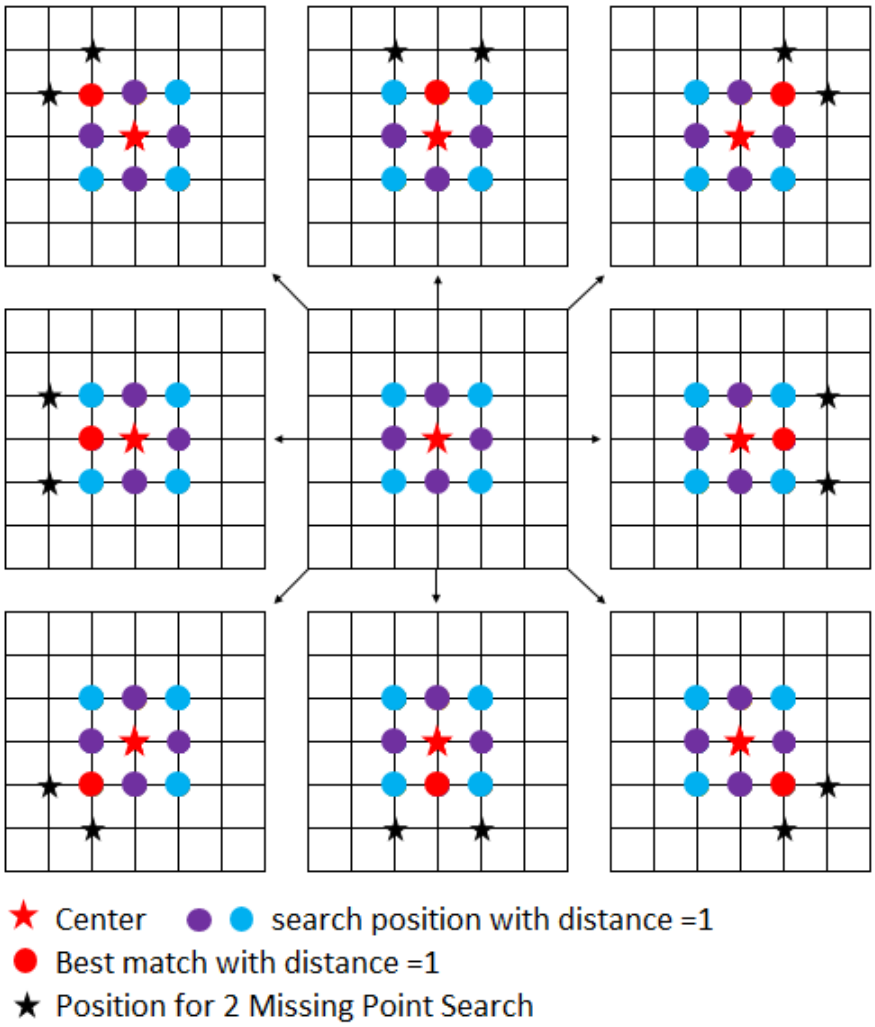


Figure 3.3: 2 Missing Points Search.

search with full search range centering at the position of the best matching block from First Search will be performed. Specially, there is an early stop criterion for the First Search. As the search order is from 1st round to 7th round with a corresponding distance, if in the r-th round ($r \in [1, 3]$) a temporary best matching block MB_x is found and in next 3 rounds, from $(r+1)$-th to $(r+3)$-th round, there is no better candidate existing, then the Refinement Search can be terminated with MB_x as its best matching block for the next stage.

2 Missing Points Search

2 Missing Point search is required only when the distance of immediate best matching is one pixels away from the search center. A distance equal to 1 implies that there are 8 possible qualified positions, depicted in the central window of Figure 3.3, of which 4 are

from the first round (marked by purple dots) and the other 4 from the 2nd round (marked by blue dots). The search patterns and strategy of 2 Missing Point search are illustrated by the 8 periphery windows for 8 cases separately, in which the red dot indicates it is the best matching position in that case and the two black stars point out the 2 Missing Points Search patters. Then after accomplishing 2 Missing Points Search the final best matching block's displacement will be stored and meantime TZ Search is finished.

Raster Search

When the distance between the start point and the best matching block from First Search is more than 5 pixels, TZ Search will enter into the Raster Search stage. The strategy of Raster Search is very simple that it is to perform a full search within the search window with interval of σ pixels in both horizontal and vertical direction, where $\sigma = 5$ in the HEVC reference software. The search patterns of Raster Search is exhibited in Figure 3.4. After the accomplishment of scanning all the search patterns, if the optimal block

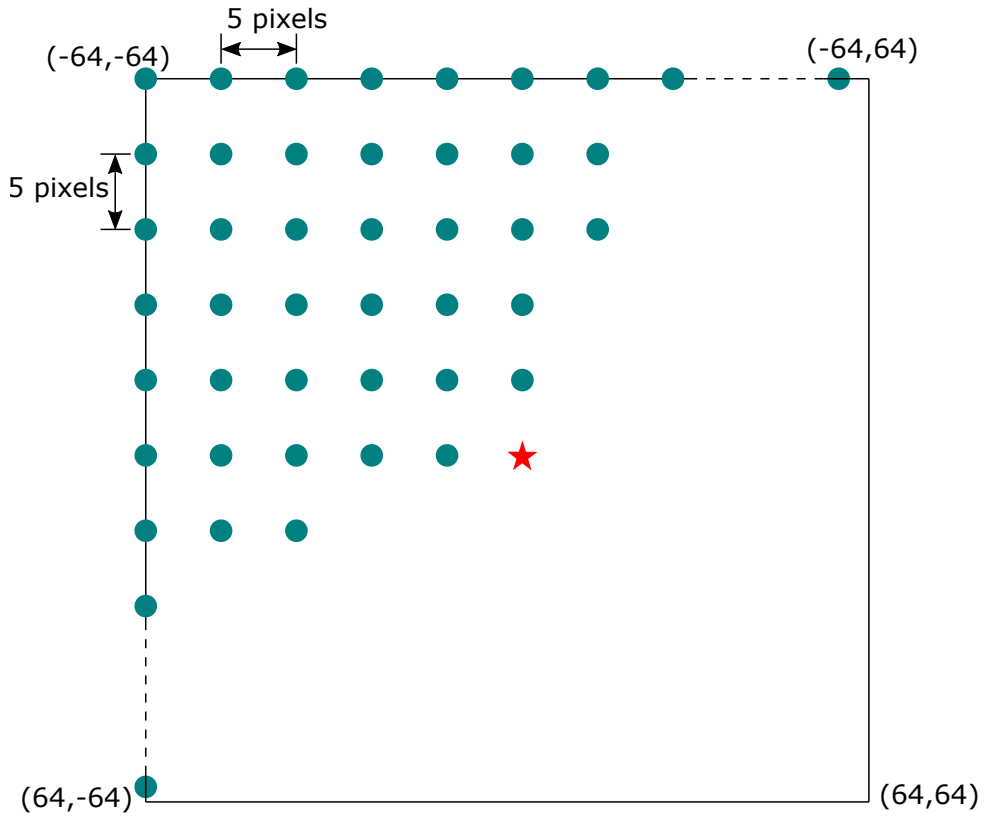


Figure 3.4: Search patterns of Raster Search with a interval of 5 pixels.

generated by Raster Search coincides with the best matching block from First Search, its displacement is regarded as the global best MV and the TZ Search can be stopped at once; or the TZ Search is continued with a refinement search of full search range.

Refinement Search

A Refinement Search is launched under two possible circumstances: (1) the best distance is between 2 to 5 pixels after the First Search; (2) the Raster Search finds an optimal matching block which is different from the result of First Search. In the TZ Search implemented in HEVC reference software, there are two options for the refinement search, one is the Star Refinement and the other is Raster Refinement. The process of Star Refinement search is indeed to fulfill a new 7-round Diamond Search (shown in Figure 3.2) with full search range taking the best matching block from last stage as the center. But it is a looped search work as illustrated in Figure 3.1. Since Star Refinement stops only when the resulting best distance is equal to zero or one, or it will continue to conduct a 7-round Diamond Search centered at the new optimal position found in the last refinement. Once the best distance is one, 2 Missing Points Search is additionally executed before stopping the TZ Search. As for the Raster Refinement, unlike Star Refinement, instead of

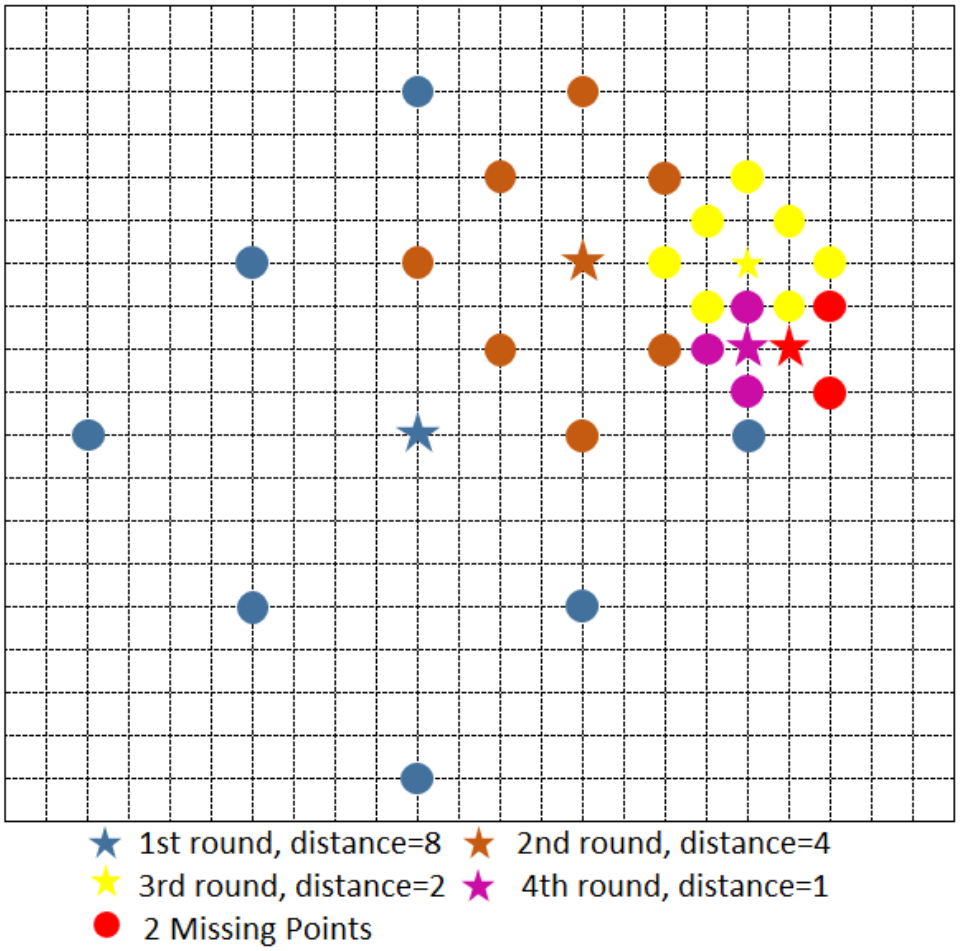


Figure 3.5: An example for Raster Refinement Search.

applying 7-round Diamond Search every time it utilizes a single-round Diamond Search

with a distance being half of the best distance of last stage. It will repeat this process and not stop until either the best distance of a certain search round equals to zero or the best distance shrinks to reach one or zero after several search rounds. An example of Raster Refinement is given in the Figure 3.5. Normally comparing to Raster Refinement, Star Refinement will search more positions because of the 7-round Diamond Search scheme, but it will definitely generate more accurate MVs and thus achieve higher video quality. As well in the HM test model, Star Refinement is set to be the default refinement scheme.

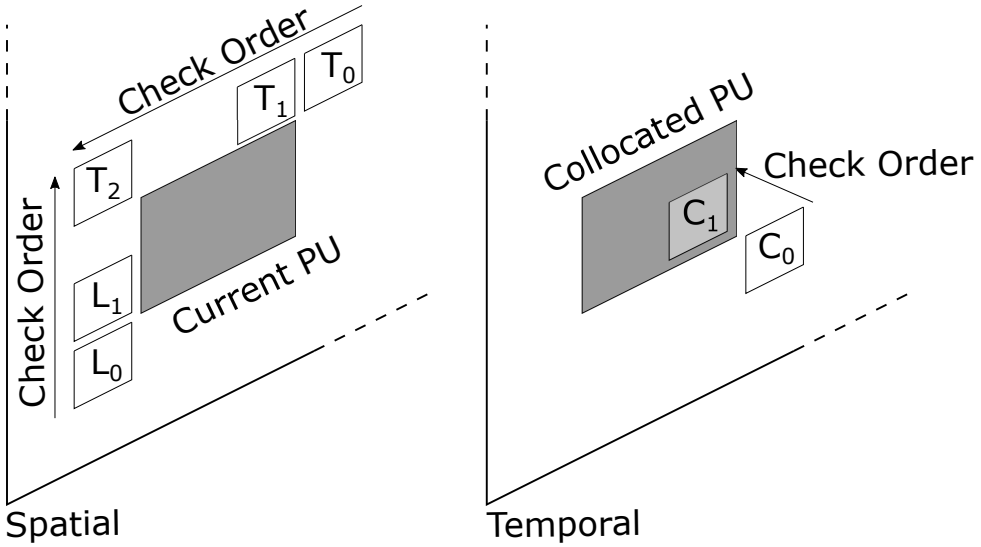
3.2 An Enhanced TZ Search using Parallelized Strategy

Although TZ search as the default fast search algorithm of HEVC reference software is a efficient algorithm with high video coding performance and reasonable complexity, there is still much redundancy existed, such as the Raster Search. Moreover, since one of the two paramount issues concerned in HEVC standard is the increased use of parallel processing architectures [1]. In this section from the aspect of parallelization two novel schemes are proposed to further enhance the TZ Search algorithm.

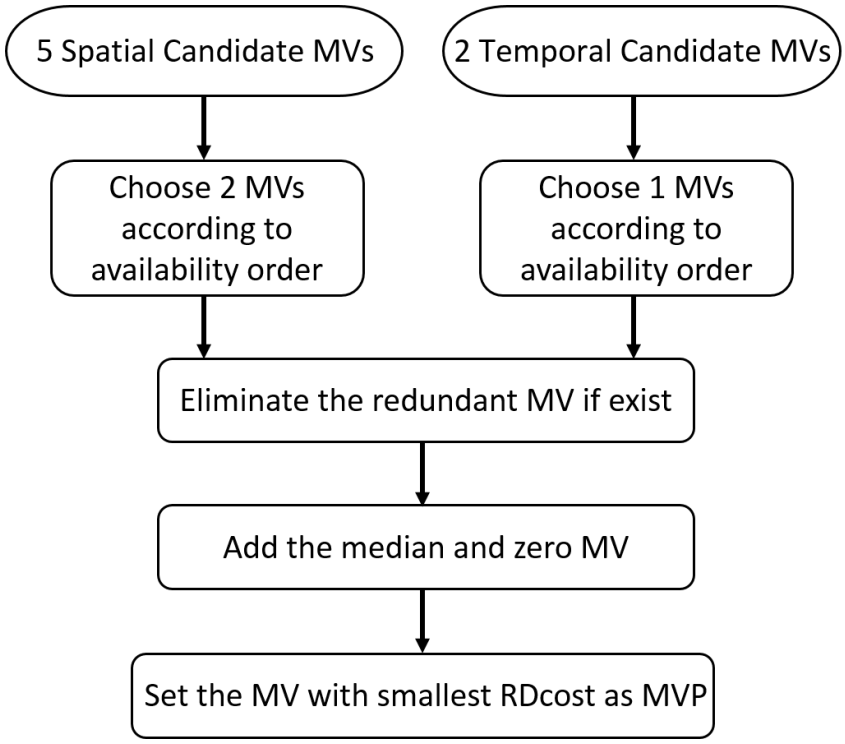
3.2.1 Multiple Initial Search Centers

As known to us, in the Raster Search it performs a full search within the whole search window with a raster of 5 pixels which is unquestionably rather time-consuming. More precisely the typical search range in HEVC standard is ± 64, it can be easily obtained that the total number of candidates for the Raster Search is $\lfloor (2\times 64+1)/5 \rfloor \times \lfloor (2\times 64+1)/5 \rfloor = 625$. Whereas a 7-round Diamond Search, considering all possible search patterns are available within the search window, will take overall $4 + 3 \times 8 + 3 \times 16 = 76$ patterns into account. Thus in order to further improve the timing performance of TZ Search, Raster Search has to be replaced.

As discussed in Section 3.1, Raster Search plays a remedy role in the TZ Search when the best matching block resulting from First Search is not accurate enough. However the accuracy of First Search relies on the quality of optimal MV predictor, namely AMVP (Advanced Motion Vector Prediction) [65–72]. Hence firstly let's have a analysis on the AMVP generating mechanism, which is presented in Figure 3.6(b). Figure 3.6(a) shows that the candidate MV list consists of spatial candidates and temporal candidates. The former one are taken from MVs of the top and left neighbors of current block. While the latter one are derived from the MVs of the collocated block and its bottom-right neighbor block in the reference frame. All of these candidate MVs are directly responsible for the



(a)



(b)

Figure 3.6: (a) AMVP MV candidates construction list. (b) The working mechanism of AMVP.

prediction accuracy of the final MVP because it is chosen from them. Taking every aspects into account, it is doubtlessly true that a MV from any position of the list is possible to be the final MVP and it is determined using the criterion of lowest distortion. However, a lowest distortion on a single position may not guarantee that the its prediction accuracy

is better than the other candidates under some circumstances. In other words, a MV candidate from the list with worse distortion may do a better prediction.

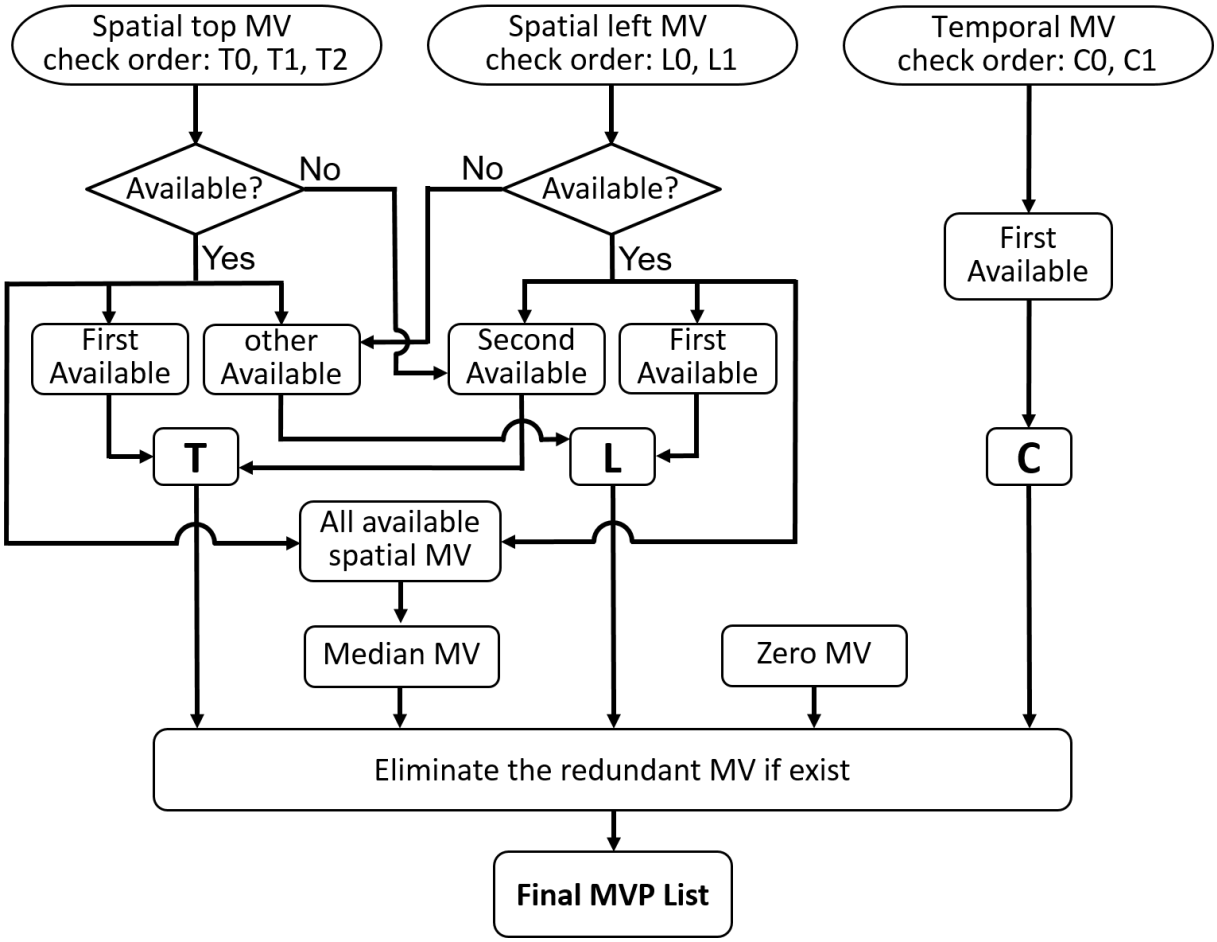


Figure 3.7: The derivation of MVP List.

Based on these observations, a strategy proposed to enhance TZ Search is taking multiple MV candidates from the list to form multiple initial search centers or start points instead of a single start point in the original TZ Search algorithm. Moreover, through some test it is found that the initial searching at multiple centers has already covered enough search patterns within the search window. Hence the Raster Search can be removed from TZ Search and it will not result in much performance degradation but can reduce the total number search points to a certain extent. The advantage of the proposed strategy is that on one hand multiple initial search centers can improve the prediction accuracy and the discarding of Raster Search can alleviate the encoder complexity. On the other hand in the point view of hardware implementation, the scheme of multiple search centers is feasible for the parallelization which can further improve the throughput of the encoder. Figure 3.7 shows the process of derivation of the MVP List, which is

generally composed of one spatial left MV, one spatial top MV, one Temporal MV, the median MV of all available spatial MVs and zero MV. Obviously it covers every category of motion vector for prediction. But in some cases if the spatial top MV is not available it will take the second available spatial left MV as a replacement and vice versa as shown in Figure 3.7. Using this new strategy, an enhanced TZ Search algorithm is produced and

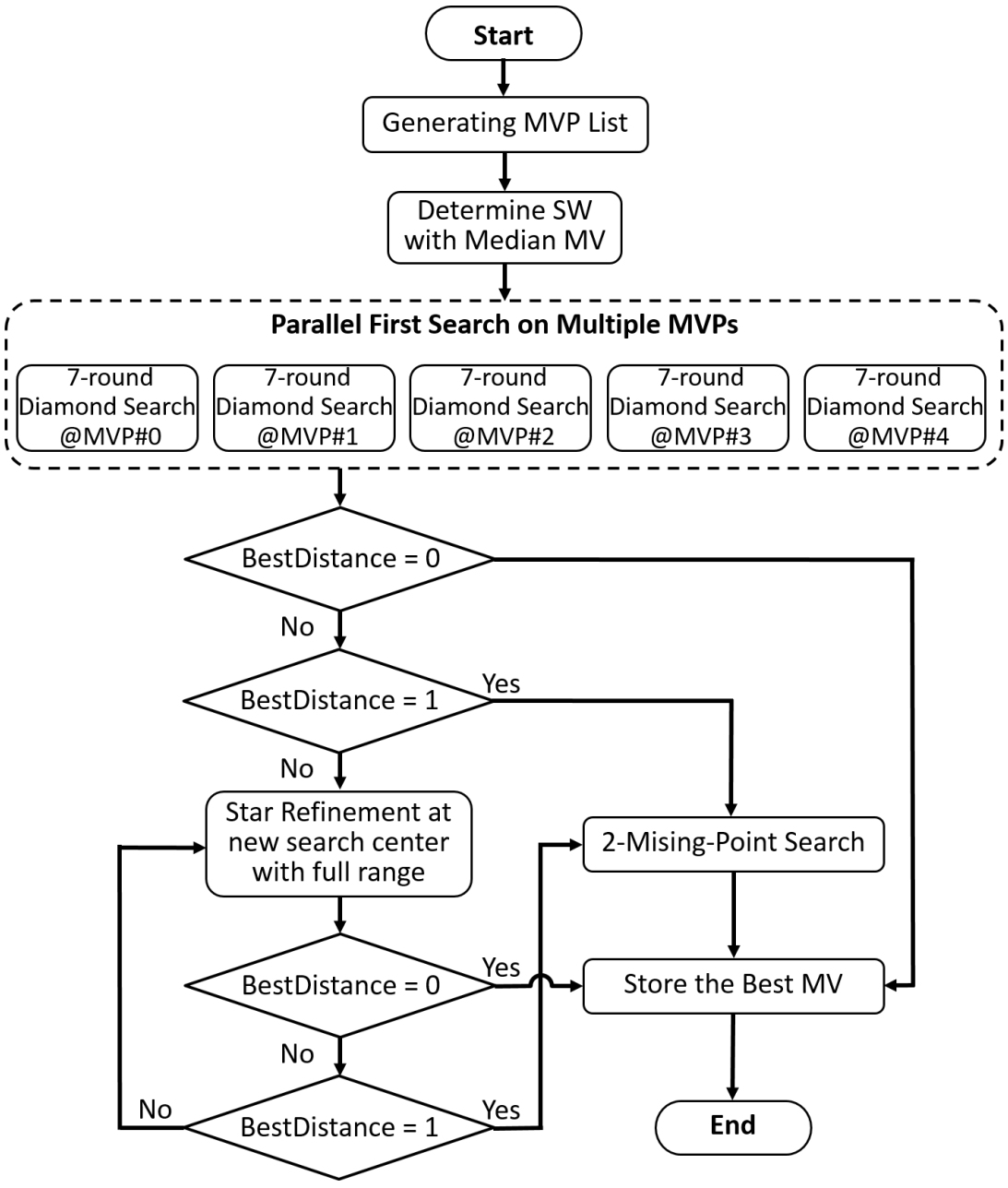
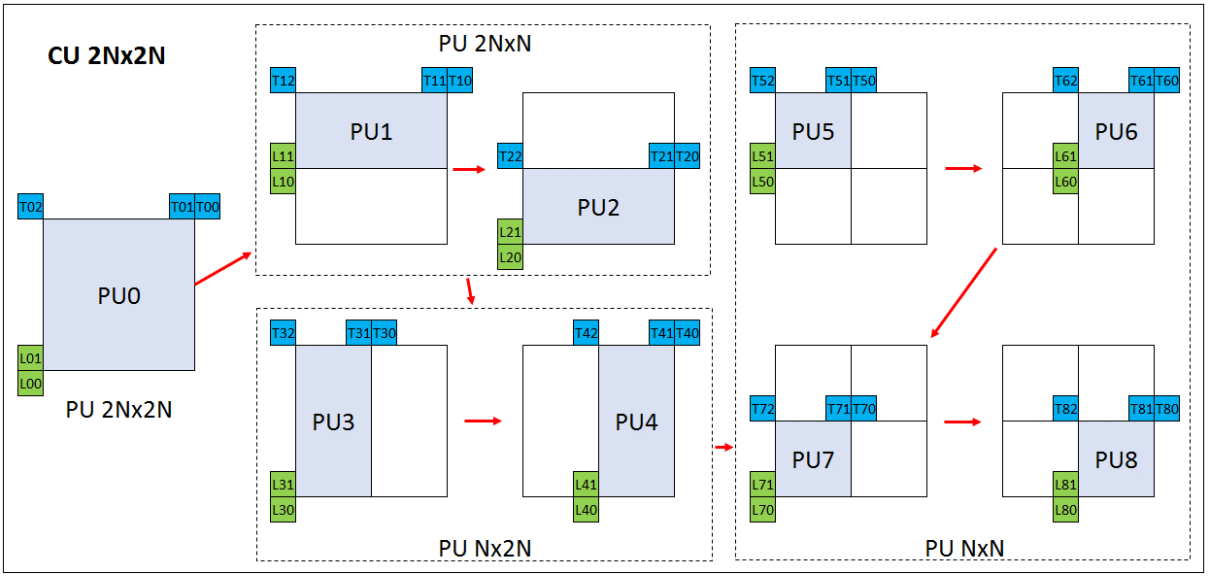


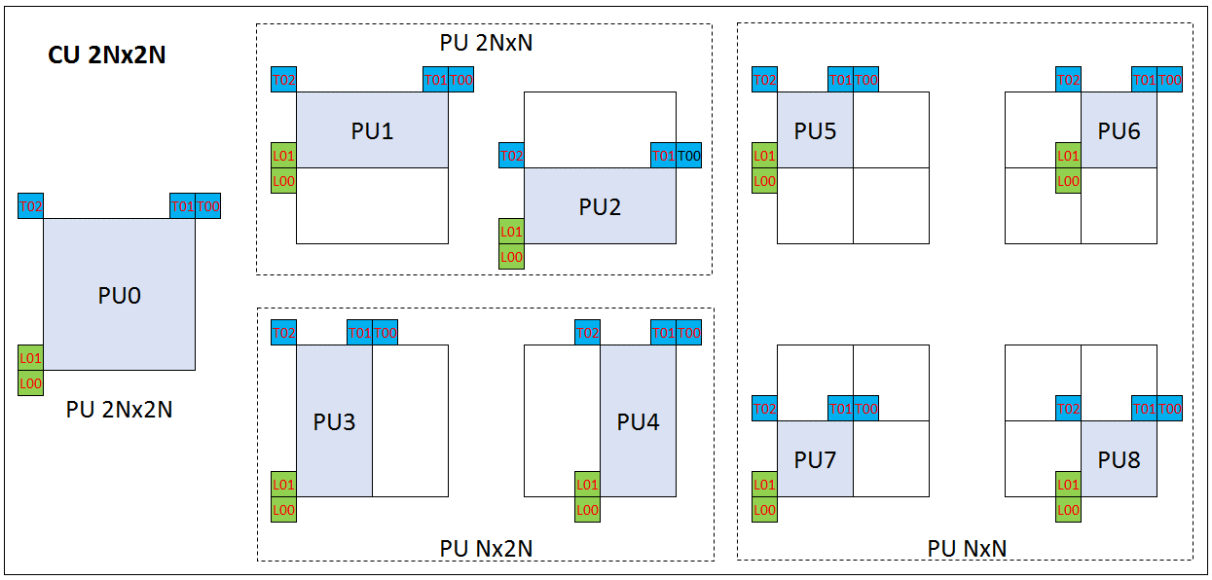
Figure 3.8: The flowchart of the enhanced TZ Search.

its working flowchart is presented in Figure 3.8. With the search window settled using the Median MV, the parallel First Search centered at the five MVPs separately will be

performed simultaneously. Except the absence of Raster Search the following procedures are remained the same with the original TZ Search which has already been discussed in Section 3.1.



(a)



(b)

Figure 3.9: (a) The default sequential processing order of PUs inside one CU. (b) Mechanism of sharing MVP between PUs of one CU.

3.2.2 Parallel ME Engines using Shared AMVP

As known to all, the main purpose of HEVC Test Model is designed for software implementation for testing novel video coding tools [1], and not specifically for hardware implementation. The TZ Search algorithm is still conducting the motion estimation for PUs from 64×64 to 8×8 sequentially, which is illustrated in Figure 3.9(a). Thus every time for a single PU, ME module needs to determine the AMVP, load the search window and then perform the best matching block searching. For the sake of improving the coding efficiency, a strategy utilizing shared AMVP on CU-level is exploited to parallelize the PU processing inside a CU. Although there is some similar work presented in literature [73], here the main contribution is on the hardware implementation. The sharing scheme is depicted in Figure 3.9(b), in which inside one CU of size 2N×2N the MVPs of PU 2N×2N, PU 2N×N and PU N×2N are all shared from PU 2N×2N. Owning to the MVP sharing scheme on CU level, the PU processing in one CU can be performed in parallel which will improve the throughput of ME module and also the timing performance. On the meantime the search window is also shared by all PUs, thus the memory time for loading frame pixels from off-chip memory to on-chip buffer will be saved to a large degree. Table 3.1 shows a comparison on ME processing rounds between HEVC reference software and proposed architecture, in which the total number of ME processing rounds of proposed TZ Search algorithm is reduced by 80% compared with the the original one in HM reference software. It is clearly that resorting to the MVP sharing mechanism, the parallelized ME engines for all PUs in one CU requires only one round of ME processing for every CU.

CU and PU in a 64×64 MB			ME Processing Rounds		
Size of CU	Num. of CU	Num. of PU per CU	HM16.2	Proposed	Savings(△)
64×64	1	5: 1 PU64×64,2 PU32×64,2 PU64×32	5	1	80%
32×32	4	5: 1 PU32×32,2 PU16×32,2 PU32×16	20	4	80%
16×16	16	5: 1 PU16×16,2 PU16×8,2 PU8×16	80	16	80%
8×8	64	5: 1 PU8×8,2 PU4×8,2 PU×4	320	64	80%
		Total	425	85	80%

Table 3.1: Design Specifications of an HEVC encoder considered in this work.

3.2.3 Hardware Implementation

Since the optimization schemes discussed in Section 3.2.1 and 3.2.2 are explored from the aspect of enhancing architecture's parallel processing capability, the key point is the hardware fulfillment. The design specifications for HEVC encoder considered in this work is presented in Table 3.2. The architectural diagram is demonstrated in Figure 3.10.

HEVC Test Model	HM16.2
Applied Prediction Structure	Low Delay P picture (bitdepth of 8)
Number of the Reference Frame	1 past frame
Search Range	±64 in horizontal and vertical direction
Maximum Supported Resolution	4096×2048 (4K videos)
Largest Coding Unit (LCU)	64×64
Supported CTU Partitioning	64×64,64×32,32×64,32×32,32×16,16×32,16×16,16×8,8×16,8×8
Target Frame Rate @4096×2048	30 fps
Target Operating Frequency	200 MHz
Process Technology for Synthesis	65-nm CMOS standard-cell technology

Table 3.2: Design Specifications of an HEVC encoder considered in this work.

We can see the ME Engine Control unit together with the Finite State Machine is responsible for the global administration of memory management, enable signals generating for the execution of functional unit, process switching between CTUs in a frame and CUs in a CTU, and final CU mode decision. These units are the same with those described detailedly in Section 2.3. Whereas the AMVP unit will generate a list of multiple MV predictors instead of a single one MVP in traditional one. In the following details on the parallel ME search engines are illustrated.

Parallel ME Search Engine

According to HEVC standard, for every CU of size 2N×2N in a 64×64 CTU (asymmetrical partitioning not considered), there are three kinds of PU: 2N×2N, 2N×N and N×2N. Moreover, there is dependency on MV between adjacent PUs which puts a limitation on the parallel processing of PUs. However with the mechanism of sharing MVP for all PUs of one CU, the parallelized architecture to find best matching block for all PUs turns out to be realizable. In Figure 3.10, three parallel architectures of ME Engine are implemented for PU 2N×2N, PU 2N×N and PU N×2N. In each of the ME engine, a First Search utilizeing parallel 7-round Diamond Search is conducted firstly, and the degree of parallelism is set to be 5 in accordance with the search strategy of multiple initial searching centers. In each of the 7-round Diamond Search, initially according to the principle of Diamond Search all available candidate's positions will be collected and stored in an array. Then the distortions between current block and all candidate block are computed sequentially using the parallel SAD calculating tree, and meantime the comparison on distortion results is performed to locate the best matching block for the current searching process. The First Search comes to the end with choosing the best MV from the 5 best candidates obtained by the 5 parallel architectures. In the next based on the result of First Search, a choice on termination of search process, 2-Missing-Point Search or Refinement

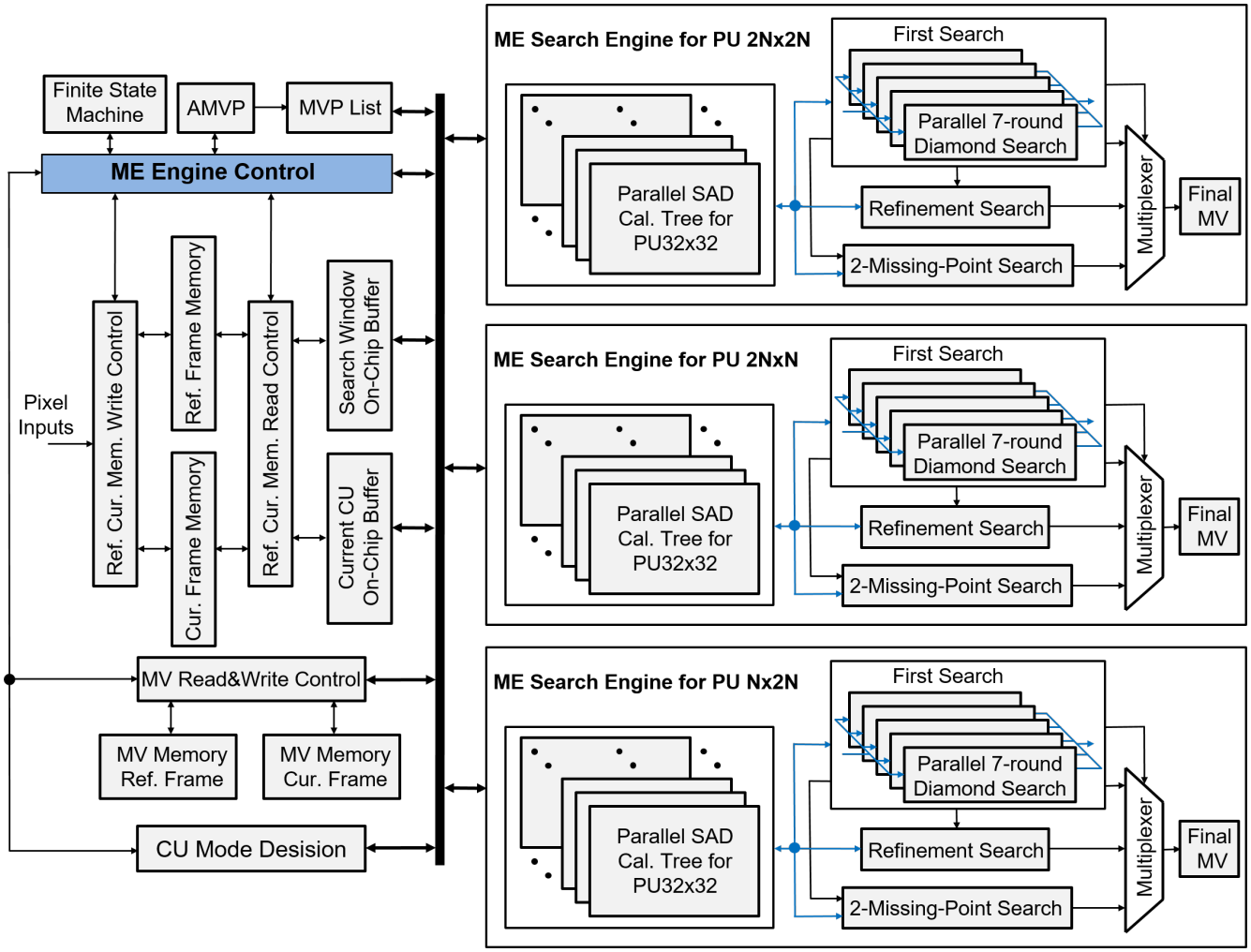


Figure 3.10: The architecture of the enhanced TZ Search.

Search will be made by a multiplexer if the distance of the best MV is 0, 1 or more than 1 respectively. At last when the final MV generated from the multiplexers is transmitted to the CU Model Decision unit, the ME processing for current CU is finished. It can be observed from the proposed architecture that the parallel processing of all PUs in one CU together with the parallelized initial searching on multiple centers will definitely improve the efficiency of the encoder appreciably.

Parallel SAD Calculating Tree

As specified in Table 3.2, the goal of the design presented in this work is to make the encoder be able to process a video in definition of 4096×2048 with a frame rate of 30fps at the target operating frequency of 200MHz. As a consequence the maximum number of clock cycles available for processing each 64×64 LCU turns out to be $(1/30)/[(4096 \times 2048)/(64 \times 64)] \times 2 \times 10^8 \approx 3255$. In order to fulfill these specification, in addition to

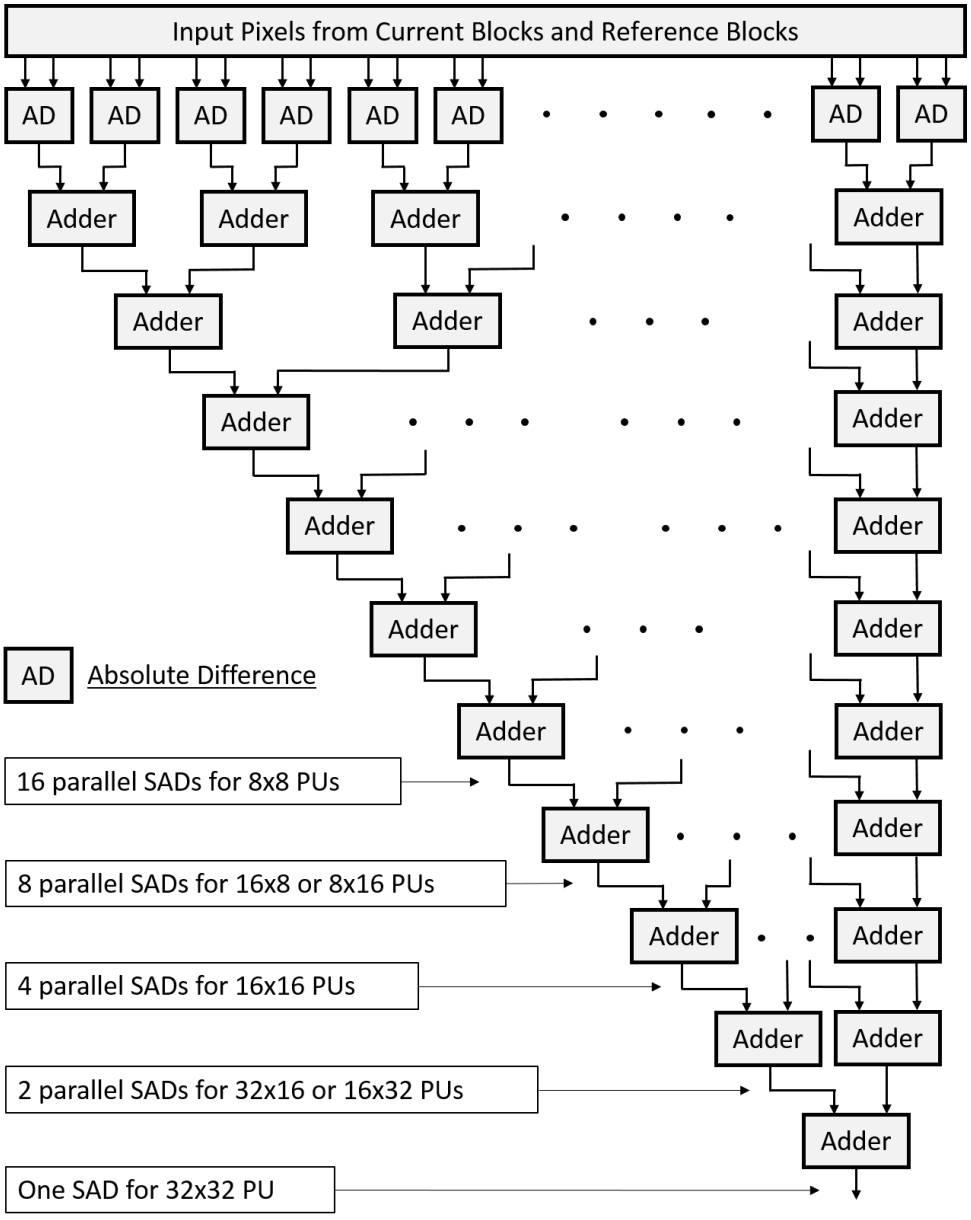


Figure 3.11: The architecture of parallel SAD calculation tree for PU32×32.

the mechanisms of parallel processing of PUs and parallel initial search, the architecture for generating SAD results is also required to be parallelized to obtain this throughput. The implementing details of parallel SAD calculating tree is demonstrated in Figure 3.11. Considering the trade-off between the efficiency and hardware cost, this architecture is capable of parallel computing the SAD between two 32×32 PUs in one clock cycle, in which overall 1024 AD (Absolute Difference) units and 1023 2-to-1 adders are utilized. For larger PU size it needs to take more than one clock cycle to get the SAD result, for instance, for PU sizes 64×64 and 64×32 (or 32×64) 4 and 2 clock cycles are expected separately. But for smaller PU size it is able to deal with multiple PUs simultaneously in

a single clock cycle. Specifically it can produce 2, 4, 8, and 16 SAD results in parallel for PUs with a size of 32×16 (16×32), 16×16, 16×8 (or 8×16) and 8×8 respectively.

Video Sequences	Algorithm	BitRate		YUV-PSNR		Encoding Time	
		kbps	△(%)	dB	△(%)	seconds	△(%)
Four People(720p)	TZ original	456.3360	–	38.8775	–	1498.37	–
	Proposed	459.1680	0.62%	38.8727	-0.01%	1474.67	-1.58%
Basketball Drive(1080p)	TZ original	3312.6600	–	36.7607	–	5142.86	–
	Proposed	3339.9920	0.83%	36.7540	-0.02%	4535.12	-11.82%
Park Scene(1080p)	TZ original	1600.6387	–	35.1279	–	4244.28	–
	Proposed	1607.9002	0.45%	35.1168	-0.03%	4097.98	-3.45%
Traffic(1600p)	TZ original	2219.4312	–	36.7072	–	7641.07	–
	Proposed	2245.3896	1.17%	36.6983	-0.02%	7493.39	-1.93%
Read Steady Go(2160p)	TZ original	11985.3600	–	39.1989	–	16055.88	–
	Proposed	12051.3600	0.55%	39.1952	-0.01%	15268.12	-4.91%
Average	Proposed	–	0.55%	–	-0.01%	–	-4.74%

Table 3.3: Simulation result using difference test sequences with QP=32.

3.2.4 Experimental Results

The proposed enhanced TZ Search algorithm is firstly implemented in the HEVC reference software. The experiment conditions is the same with what is specified in Section 2.2.3. The simulation results using 5 different high definition test sequences and a QP value of 32 is listed in Table 3.3. Compared to the original TZ Search, the proposed one improves the total encoding time by 4.74% with less than 0.6% bit-rate increase and 0.01% PSNR degradation. One thing needs to be clarified that the proposed algorithm implemented in HM test model is not applied with any parallel processing schemes but only with the utilization of multiple initial search centers and removal of the Raster Search process. Thus hardware architecture implemented with proposed strategies and parallelization mechanisms will definitely gain a much better performance improvement.

3.3 Summary

In this chapter, an enhanced TZ Search algorithm is explored with new strategy of multiple initial search centers and shared MVP on CU-level. Moreover because of the better prediction of multiple-MVP the Raster Search scheme is disabled for reducing the number of search positions. The results of simulation with HM test model demonstrates it achieves a reduction of about 5% on the whole encoding time with negligible bit-rate increment and video quality degradation. Additionally the hardware implementation of the proposed architecture employs several parallelization mechanisms including parallel ME engine for

PUs within a CU based on the shared MVP strategy, parallel initial searching centered at multiple MVPs, and the parallelized SAD calculating tree for PU size 32×32. Thus with these parallelized components the proposed architecture is expected to obtain an appreciable improvement on throughput and encoding efficiency.

Chapter 4

High Speed VLSI architecture for finding the first W maximum/minimum values

4.1 Introduction

Nowadays along with the increasing demands of high data-rate, high integration and small physical size communication systems, design of efficient digital architectures being able to accelerate key components of the system is of high significance. This work addresses the design exploration of a digital architecture dedicated to a specific task: finding N maximum/minimum values out of M inputs with high speed and low complexity.

Sorting is a well-established problem in computer science [74] and is a key operation in several applications. Besides, hardware implementation of sorting networks has been addressed as well in the past [74]. On the other hand, VLSI architectures for partial sorting, which can also be derived from selection networks (SN) [75], are part of different algorithms in the communication field. Partial sorting is employed, for example, in [76, 77] and [78, 79] for the decoding of turbo and binary Low-Density-Parity-Check (LDPC) codes, in [80] for maximum-likelihood decoding of arithmetic codes and in [81–83] forK-best MIMO detectors, non-binary LDPC decoders and turbo product codes respectively. Circuits for finding the first two minimum values, are used in binary LDPC decoder architectures [84–86] to implement min-*sum* approximations [79, 87] and recently they have also been proposed for the case of non-binary LDPC decoders [88]. However, very few works, e.g. [89, 90], investigate the general problem of implementing parallel architectures for finding the first two maximum/minimum values with a systematic approach. Similarly, architectures for finding the first $W > 2$ maximum/minimum values in a set of M elements, with $W \leq M/2$, are designed in VLSI implementations of i) K-best MIMO

detectors [91, 92], ii) non-binary LDPC decoders [93, 94], iii) turbo product codes [95]. Unfortunately, to the best of our knowledge, no papers in the open literature present a general analysis for the case $W > 2$. In this chapter two kinds of VLSI architectures for for finding the first W maximum/minimum values is exploited towards improving the speed and minimizing the hardware cost.

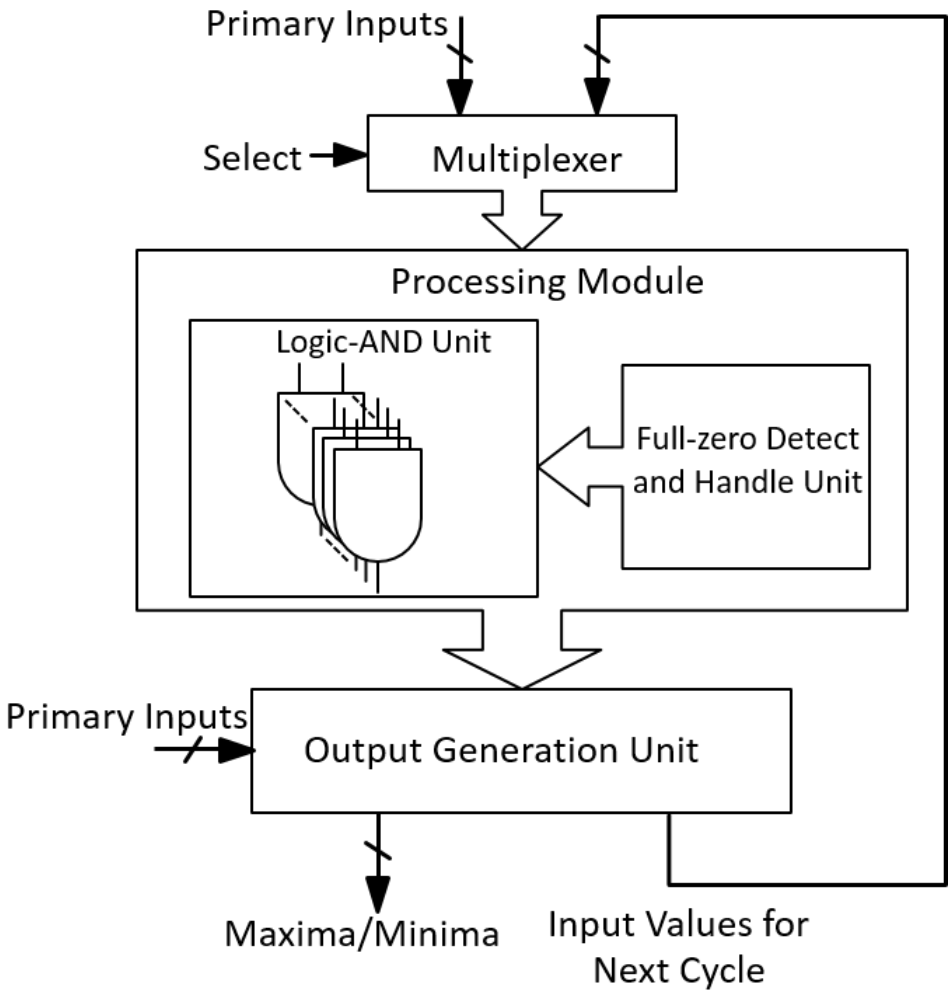


Figure 4.1: Block diagram of BWA architecture.

4.2 A Radix-sort-based VLSI Architecture with Low Cost[1]

Stemming from the work described in [75] for sorting networks, in [74] a comparator-based SN is proposed. However, as argued in [74], other approaches, such as the one referred to as radix sorting, can be used as well. Compared to the conventional sorting architectures, which are mainly based on comparators, radix sorting algorithms rely on the bit-wise analysis of the data to be sorted and can be extended to selection and partial sorting problems. Thus in this kind of architectures comparison between values is totally discarded, but instead only with some simple logic operations. This work proposes a parallel VLSI architecture relying on the radix sorting approach for finding the first $W > 2$ maximum/minimum values in a set of M values. Namely, the proposed solution, referred to as Bit-Wise-And (BWA) architecture, works by analyzing the M candidates from the Most-Significant-Bit (MSB) to the Least-Significant-Bit (LSB). The block diagram of the BWA architecture is depicted in Figure 4.1. In the following sections, details of the architecture are introduced.

4.2.1 Problem formulation and the BWA Algorithm

According to [90], we can state the problem of finding the first W maximum/minimum values as follows. Given a set $\mathcal{X}^{(M)} = \{x_0, \ldots, x_{M-1}\}$ made of M elements, we want to find the first W maximum/minimum values, namely $\mathbf{y}^{(M)} = \{y_0^{(M)}, y_1^{(M)}, \ldots, y_q^{(M)}, \ldots, y_{W-1}^{(M)}\}$ where $y_0^{(M)} = \max(\mathcal{X}^{(M)})$, $y_1^{(M)} = \max(\mathcal{X}^{(M)} \backslash \{y_0^{(M)}\})$, $\ldots$, $y_q^{(M)} = \max(\mathcal{X}^{(M)} \backslash \bigcup_{k=0}^{q-1}\{y_k^{(M)}\})$, $\ldots$, $y_{W-1}^{(M)} = \max(\mathcal{X}^{(M)} \setminus \bigcup_{k=0}^{W-2}\{y_k^{(M)}\})$ (similarly substituting max with min). For the sake of simplicity in the following we will discuss only the max case.

Radix sorting relies on the analysis of $\mathcal{X}^{(M)}$ values bit by bit from the MSB to the LSB. In the following, for the sake of simplicity, we will assume that the values in $\mathcal{X}^{(M)}$ are non-negative. It is worth noting, that 2's complement values can be straightforwardly used as well. Indeed, any set of N-bit 2's complement values can be converted in non-negative values, preserving the order relation, by flipping the MSB. Thus, let $x_a = \{x_{a,N-1}x_{a,N-2}\ldots x_{a,1}x_{a,0}\}$ and $x_b = \{x_{b,N-1}x_{b,N-2}\ldots x_{b,1}x_{b,0}\}$ be two N-bit non-negative binary values, where $x_{a,j}$ and $x_{b,j}$ represent the j-th bit of x_a and x_b respectively.

[1]Content of this section is partly from the paper: Guoing XIAO, Maurizio MARTINA, Guido MASERA, Gianluca PICCININI, "A parallel radix-sort-based VLSI architecture for finding the first W maximum/minimum values", IEEE Transactions on Circuits and Systems II, vol. 61, issue 11, pp.890-894, Aug 2014, DOI:10.1109/TCSII.2014.2350333

Assuming the first $(N-j-1)$-th MSBs of x_a and x_b have the same value, we can easily obtain the relationship between x_a and x_b based on bit-wise analysis: $x_a > x_b$ if $x_{a,j} =$ '1' and $x_{b,j} =$ '0', and viceversa. The proposed BWA relies on performing recur-

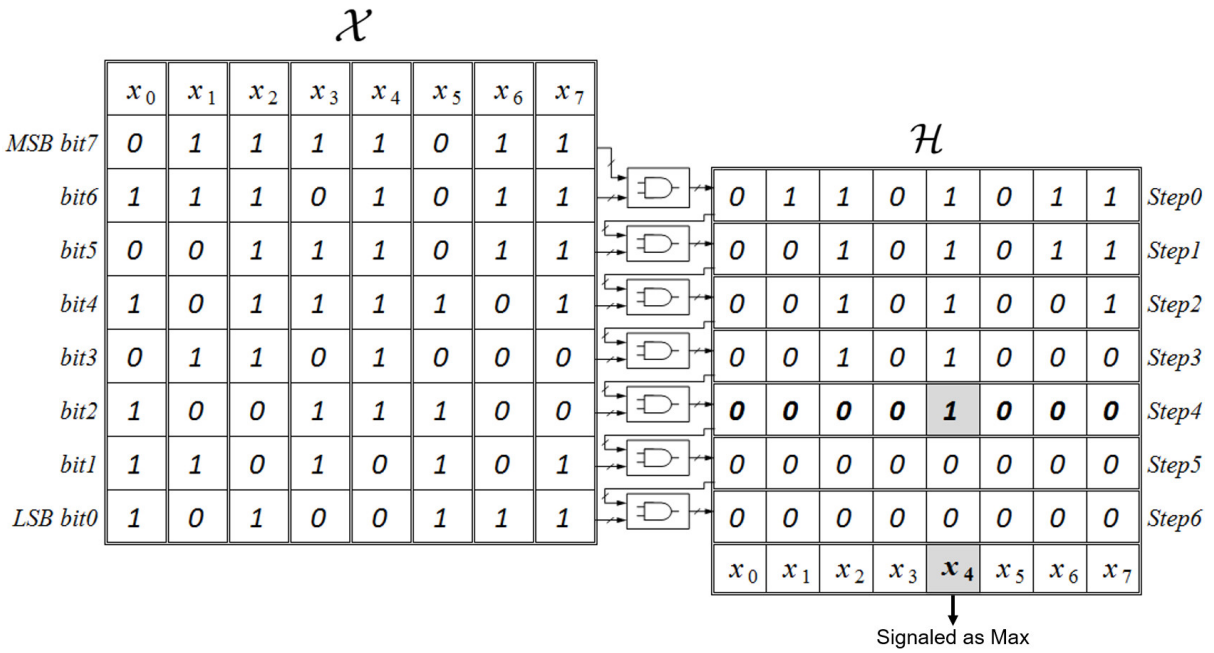


Figure 4.2: An example to exhibit working principle of BWA algorithm.

sively the logic-and operation between adjacent bits of each x_i value from the MSB to the LSB. Let $h_i = \{h_{i,N-2} \dots h_{i,0}\}$ be the array of bit-wise logic-and operations on x_i, where $h_{i,N-2} = x_{i,N-1} \wedge x_{i,N-2}$ and

$$h_{i,j} = h_{i,j+1} \wedge x_{i,j}, \tag{4.1}$$

for $j = N-3, \dots, 0$ with $\wedge$ representing the logic-and operation. If the MSB of all the x_i values is '1' and all the x_i are *monotonic* sequences of bits, that is only a transition from '1' to '0' is allowed as in the four x_i values of Example 1,then, analyzing the content of h_i for $i = 0, \dots M-1$ from the LSB to the MSB allows to find the first W maximum values.

Example 1

$$\begin{aligned} x_0 &= \{1\ 1\ 1\ 1\} \\ x_1 &= \{1\ 1\ 0\ 0\} \\ x_2 &= \{1\ 0\ 0\ 0\} \\ x_3 &= \{1\ 1\ 1\ 0\} \end{aligned} \qquad \mathcal{H} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

More specifically, let's take the set $\mathcal{X}^{(M)}$ with $M = 8$ and each of 8-bit. Figure 4.2 shows the general working principle of BWA algorithm, in which the $\mathcal{H}$ matrix generated by

logic-and operation performed from the MSB-row to the LSB-row contains a row with a single '1' in Step 4 which indicates x_4 as the maximum.

As an example, $h_{i,0}$ = '1' if and only if $x_{i,j}$ = '1' for every $j = 0, \ldots, N-1$, namely $x_i = 2^N - 1$. Let $\mathcal{H}$ be the $(N-1) \times M$ matrix whose columns are h_i. If $\mathcal{X}^{(M)}$ contains only distinct elements: i) moving from the MSB-row to the LSB-row all rows of $\mathcal{H}$ are different up to a certain j, then for $j' < j$ all the rows are zero ($j = 0$ in Example 1), ii) when moving from the LSB-row to the MSB-row, after the first non-zero row, one additional '1' appears along a column. As a consequence, moving from the LSB-row to the MSB-row of $\mathcal{H}$, the columns of the first W non-zero rows are the positions of the first W maximum values. Since in general x_i is not a monotonic sequence and repeated elements can exist in $\mathcal{X}^{(M)}$, modifications to effectively employ the BWA technique are required.

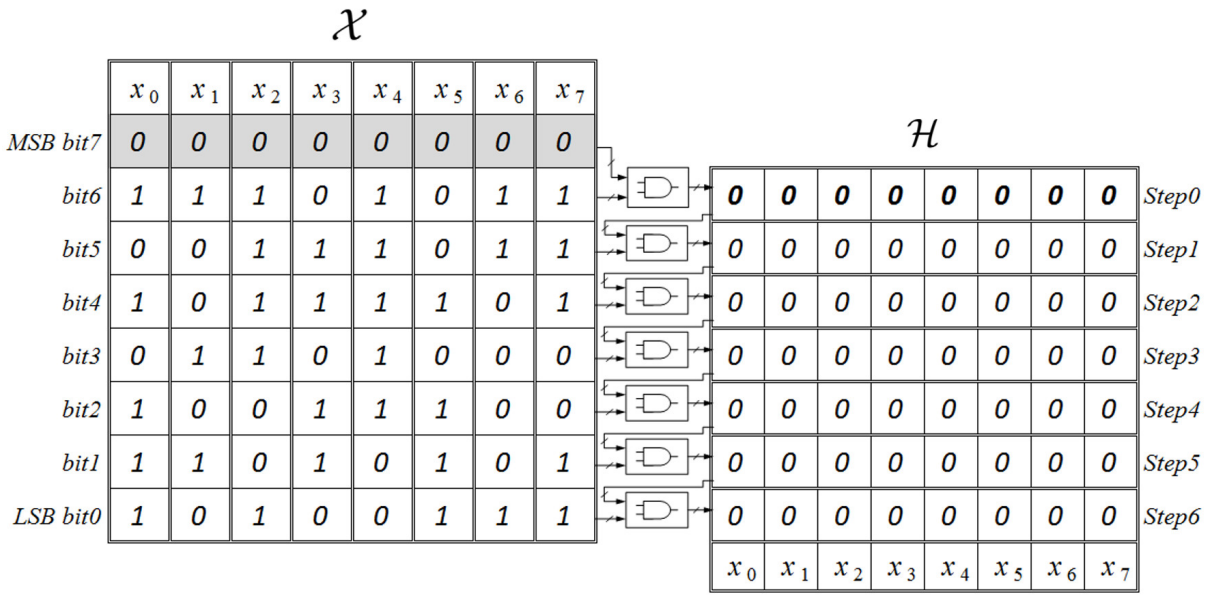


Figure 4.3: A special case #1 for showing the failure of the initial BWA principle.

4.2.2 Completed BWA Architecture

Logic-AND unit

As highlighted in Section 4.2.1, the initial BWA principle can be employed on data that are monotonic sequences of bits whose MSB is '1'. If the data in $\mathcal{X}^{(M)}$ do not meet this requirement, the architecture does not work correctly. As an example, the case $x_{i,j}$ = '0' for a certain j and for every $i = 0, \ldots, M-1$ causes $h_{i,j'}$ = '0' for every $j' \leq j$. In this case, the architecture can not distinguish among different x_i. A similar problem arises

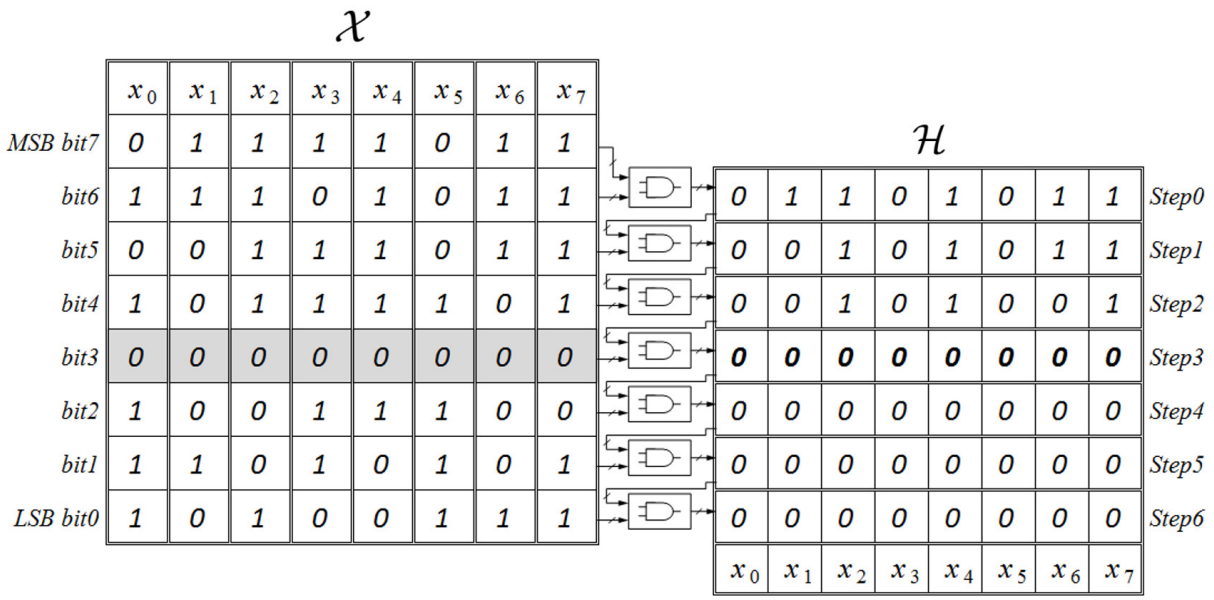


Figure 4.4: A special case #2 for showing the failure of the initial BWA principle.

if two or more x_i values are non-monotonic sequences of bits. In order to have a clear understanding of this situation, four different particular real cases given in Figure 4.3-4.6 illustrate under what kind of circumstances the initial BWA principle will fails to locate the maximum target. These cases in the figures refer to: (1) MSBs of all inputs equal to '0', (2) the bits on the same position of all inputs equal to '0', (3) results of logic-and operation between the first two MSBs equal to '0' and (4) a certain row of the logic-and

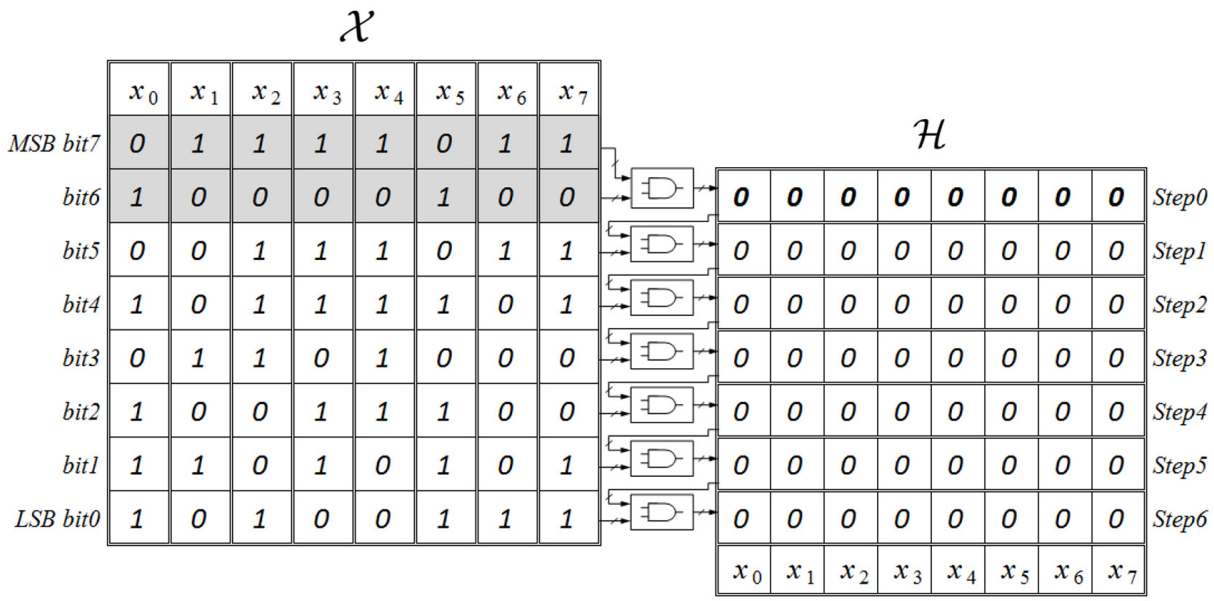


Figure 4.5: A special case #3 for showing the failure of the initial BWA principle.

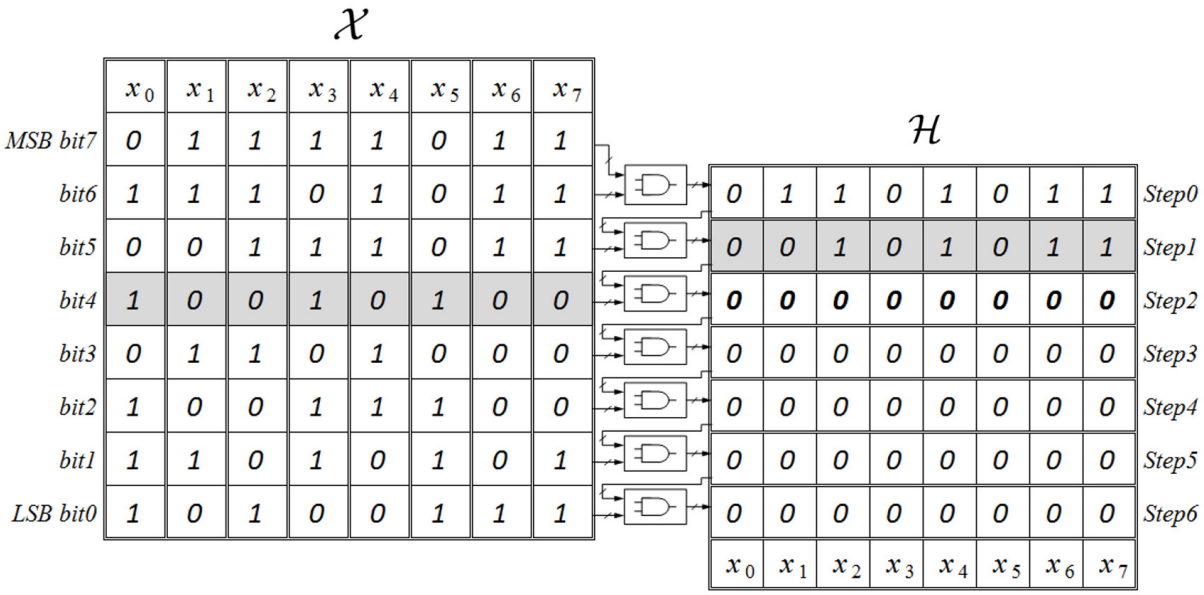


Figure 4.6: A special case #4 for showing the failure of the initial BWA principle.

results together with the next row of bits of inputs generates full-zero row, respectively.

Full-zero Detect and Handle Unit

In order to avoid the aforementioned situation, we add some gates to handle these cases, referred to as *zero-row conditions*. To this purpose we modify (4.1) as $h_{i,j} = \hat{h}_{i,j+1} \wedge x_{i,j}$ where

$$\hat{h}_{i,j} = \begin{cases} z_{N-1} \vee x_{i,N-1} & \text{if } j = N-1 \\ (z_j \wedge \hat{h}_{i,j+1}) \vee h_{i,j} & \text{if } 0 \leq j < N-1 \end{cases}, \tag{4.2}$$

$\vee$ is the logic-or operation and

$$z_j = \begin{cases} \text{not} \left(\bigvee_{i=0}^{M-1} x_{i,N-1} \right) & \text{if } j = N-1 \\ \text{not} \left(\bigvee_{i=0}^{M-1} h_{i,j} \right) & \text{if } 0 \leq j < N-1 \end{cases} \tag{4.3}$$

detects a zero-row condition. Follows an example.

Example 2

$$\begin{array}{ll} x_0 = \{0\ 1\ 1\ 1\} & z_3 = 1 \\ x_1 = \{0\ 1\ 0\ 1\} & z_2 = 0 \\ x_2 = \{0\ 0\ 0\ 0\} & z_1 = 0 \\ x_3 = \{0\ 1\ 1\ 0\} & z_0 = 0 \end{array} \qquad \hat{\mathcal{H}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

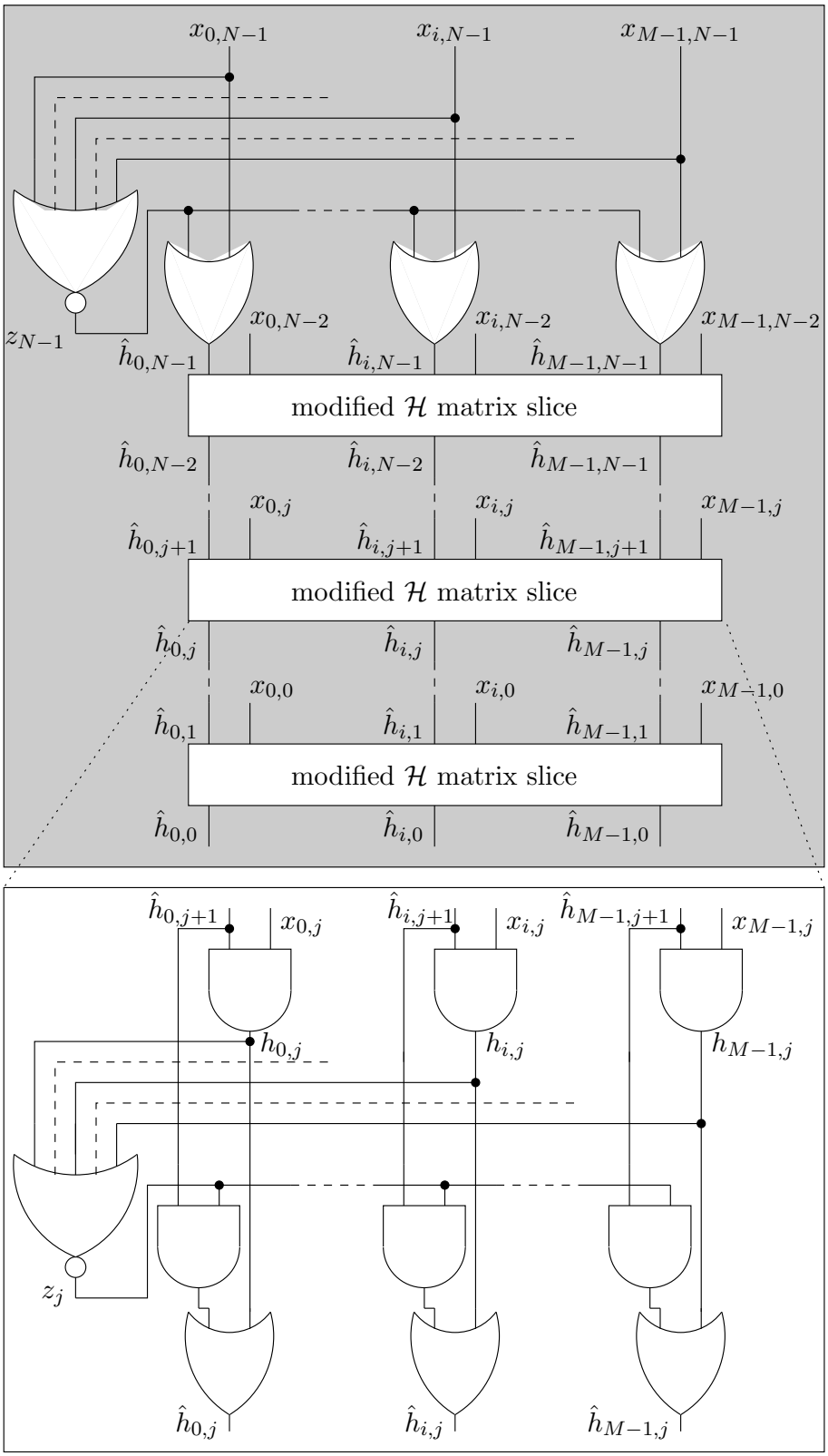


Figure 4.7: BWA architecture: modified $\mathcal{H}$ matrix.

Example 2 shows a simple case, where the modified $\mathcal{H}$ matrix ($\hat{\mathcal{H}}$), that is an $N \times M$ matrix, is given. Indeed, as explained in the next paragraphs, the maximum values are selected checking $\hat{h}_{i,0}$ values.Handling zero-rows leads to the slice-architecture depicted in light gray in Figure 4.7, where each slice corresponds to one row of $\hat{\mathcal{H}}$. The bottom part of Figure 4.7 shows the circuit to implement (4.2) and (4.3), where $\hat{h}_{i,j}$ acts as $h_{i,j}$, but, if a zero-row condition occurs, then $\hat{h}_{i,j} = \hat{h}_{i,j+1}$. As it can be observed in the modified $\mathcal{H}$ matrix, the proposed structure ensures $\hat{h}_{i,0}$ = '1' for at least one value of $i = 0, \ldots M-1$. Thus, the selection of the maximum values in the proposed architecture is performed checking $\hat{h}_{i,0}$ values. Let $\mathcal{I}$ be the set of indices $i = 0, \ldots, M-1$ such that $\hat{h}_{i,0}$ = '1'. If $\mathcal{I} = \{k\}$, i.e. it contains only one element, then $y_0 = x_k$. Otherwise, $\mathcal{X}^{(M)}$ contains more instances of the maximum value. If $\mathcal{I}$ contains W elements, then the first W maximum values are the elements $\{x_i : i \in \mathcal{I}\} \subset \mathcal{X}^{(M)}$. If $\mathcal{I}$ contains less than W elements a new search is required. As Figure 4.3-4.6 exhibits the four failure cases of initial BWA architecture, now with the Full-zero handling circuit we will present how the BWA architecture manage to locate the maximum value successfully for these cases. The case with MSBs of all input equal to '0' is handled in Figure 4.8, in which a N-to-1 NOR gate takes the zero-row of MSB as input and produces a logic-one result. In the next the generated logic-one will prevent the signal of zero-row from flowing forward resorting to the OR gate so that the following logic-and gates can operate to obtain the correct result. Similarly, in Figure 4.9 it shows in one certain stage of the logic-and operation when its result is full-zero the big N-to-1 NOR gate will generate a logic-one signal for the following logic-and gate array and logic-or gate array to maintain the last valid logic-and result for the next logic-and operation and avoid the appearance of full-zero result. The other two cases shown in 4.4 and 4.5 can also be manipulated by the same mechanism. Meantime for the sake of having a more clear knowledge of the whole BWA architecture, a full image of the architecture with a real case to find the maximum value from a set $\{x_0, x_1, x_2, x_3\}$ is illustrated in Figure 4.10.

Output Generation Unit

To simplify the selection we use a circuit referred to as *output generation circuit* that, based on $\hat{h}_{i,0}$ values, is able to find the maximum among M elements and to produce a new set of M elements $\mathcal{X}' = x'_0, \ldots, x'_{M-1}$ where the maximum value is replaced by zero.

Thus, the complete architecture, shown in Figure 4.12, is made of W stages, where each stage contains one instance of the circuit to produce the modified $\mathcal{H}$ (light gray part) and one instance of the output generation circuit (dark gray part). As a consequence, the q-th stage finds $y_q^{(M)}$, that corresponds to the maximum value of the q-th input set. This operation is accomplished by the means of the output generation circuit shown in dark

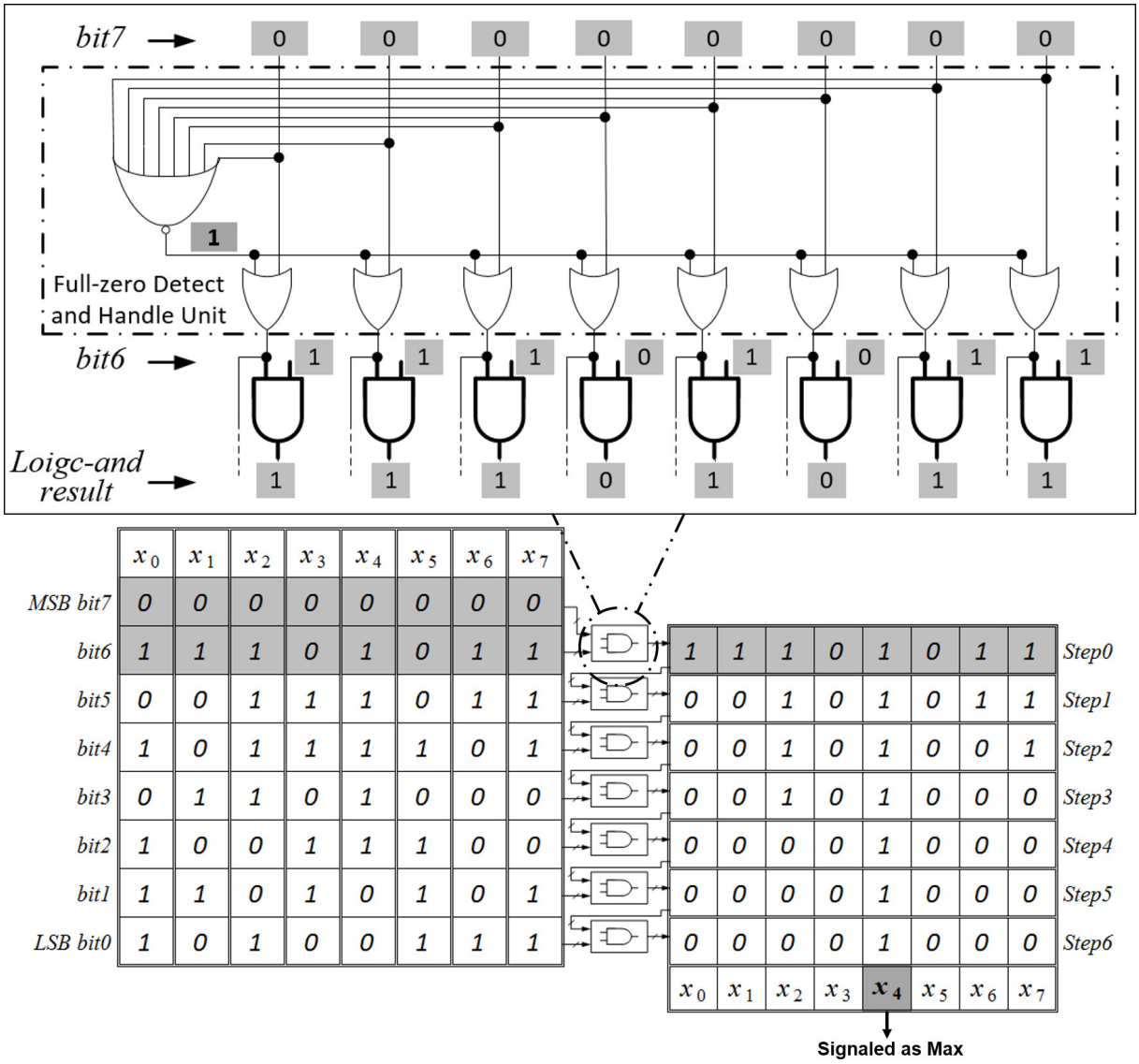


Figure 4.8: The working principle of Full-zero Detect and Handle Unit for special case #1 in Figure 4.3.

gray in Figure 4.11 for the case $q = 0$. The output generation circuit relies on $M - 1$ blocks referred to as g slice, $M \times N$ selection blocks and N combiners each made of an M-input logic-or, where M selection signals $g_i = \text{not}(\tau_{i-1}) \wedge \hat{h}_{i,0}$ for $i = 1, \ldots, M-1$ and $g_0 = \hat{h}_{0,0}$ are used in the selection blocks to forward x_i to the next slice or to replace it by zero. Each g_i signal is implemented as a slice (see the top left part of Figure 4.11), where the τ_{i-1} term is obtained as

$$\tau_{i-1} = \bigvee_{u=0}^{i-1} \hat{h}_{u,0}, \tag{4.4}$$

that is: when $\hat{h}_{u,0} =$ '1', then the remaining τ_l with $l = u + 1, \ldots, M - 1$ are '1' and so

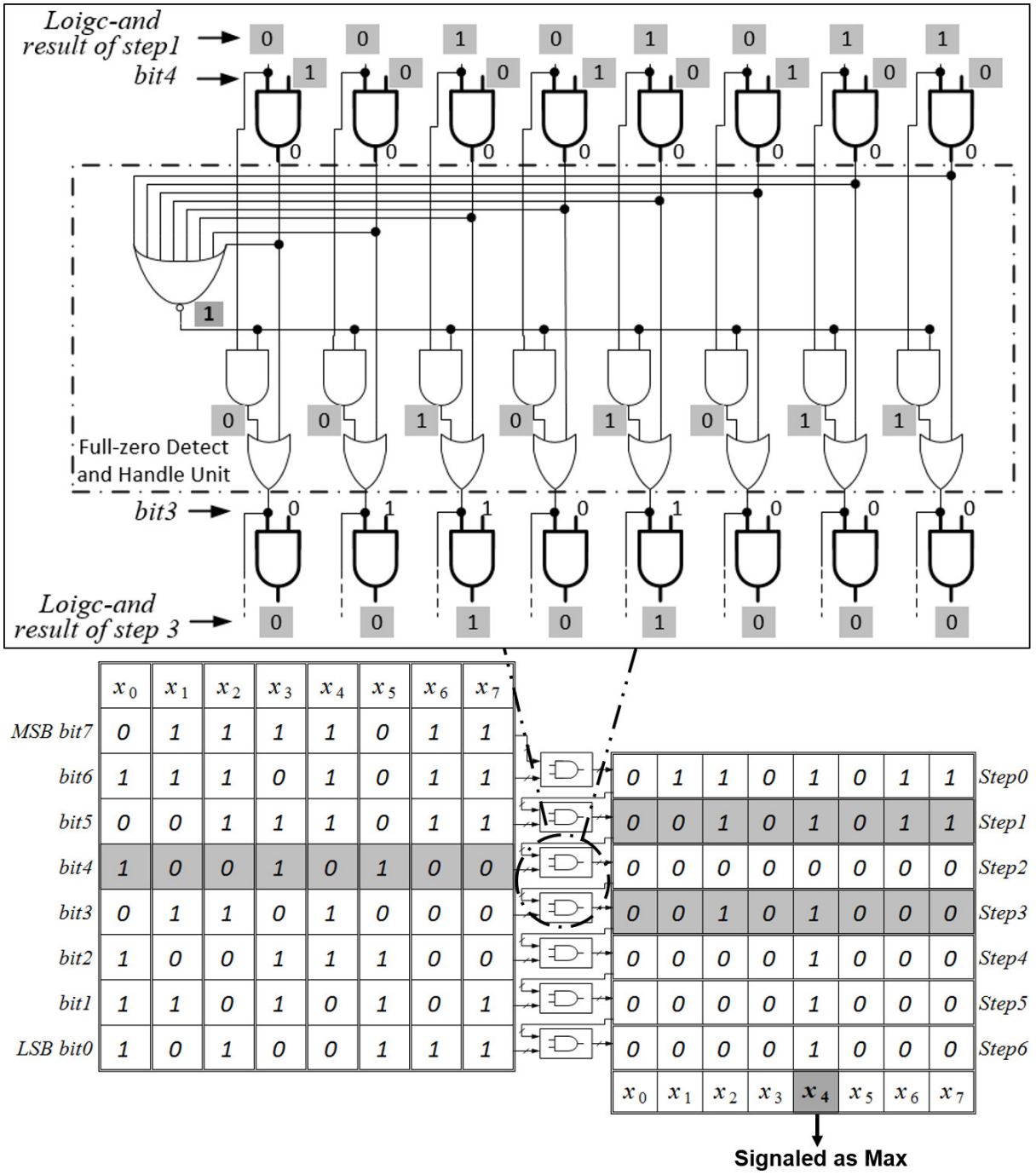


Figure 4.9: The working principle of Full-zero Detect and Handle Unit for special case #4 in Figure 4.6.

in the current stage only x_u is selected. More precisely, with reference to Figure4.11, g_i is exploited in the selection blocks to compute $\zeta_{q,i}$, one of the M candidates, where only

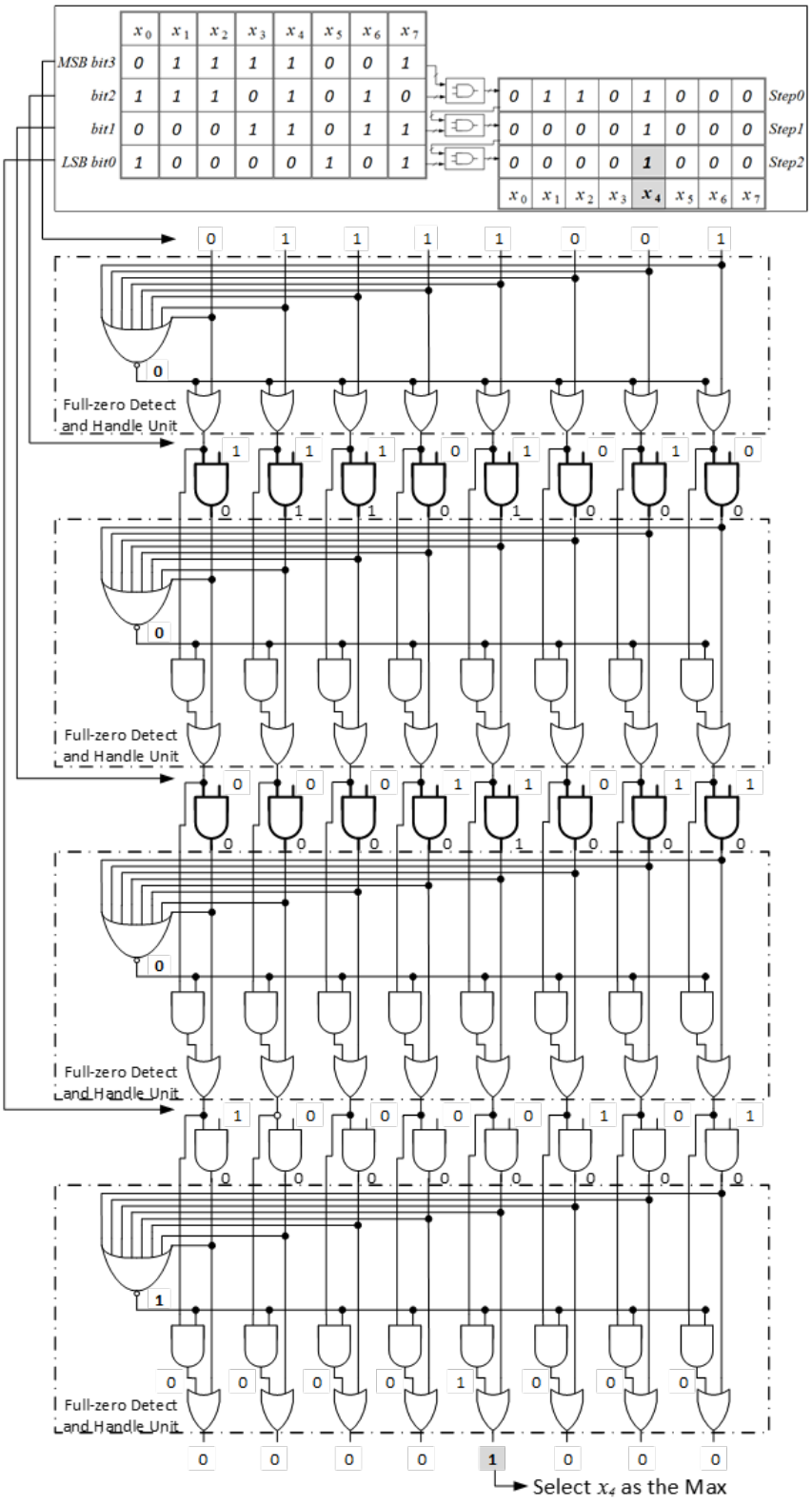


Figure 4.10: An example for the full BWA architecture.

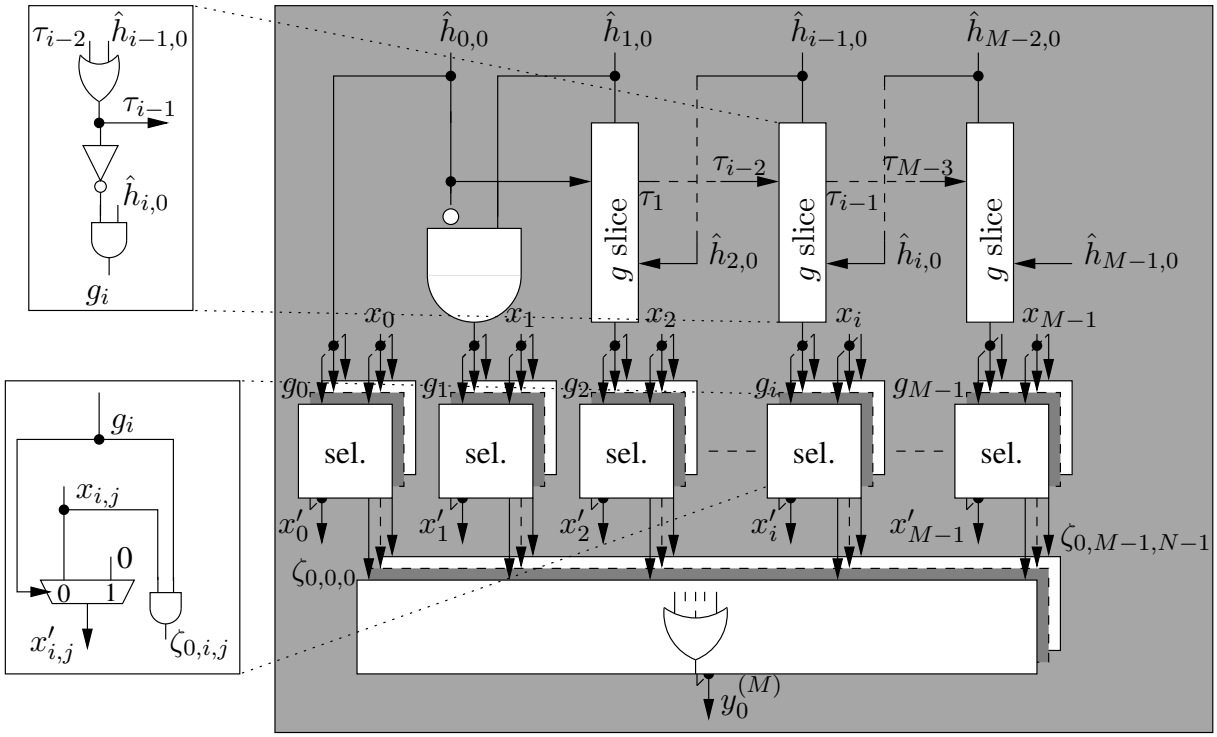


Figure 4.11: BWA architecture: output generation circuit in the case $q = 0$.

one $\zeta_{q,i} \neq 0$. Each bit of $\zeta_{q,i}$ is computed as $\zeta_{q,i,j} = g_i \wedge \chi_{q,i,j}$ where

$$\chi_{q,i} = \begin{cases} x_i & \text{if } x_i \notin \bigcup_{k=0}^{q-1} \{y_k^{(M)}\} \\ 0 & \text{otherwise} \end{cases}, \tag{4.5}$$

and the terms $y_{q,j}^{(M)}$ and $\chi_{q,i,j}$ represent the j-th bit of $y_q^{(M)}$ and $\chi_{q,i}$ respectively (see the sel. block in the left part of Figure 4.11). As an example, for $q = 0$ and $q = 1$ we have $\chi_{0,i} = x_i$ and $\chi_{1,i} = x'_i$ respectively. Finally, the q-th maximum is obtained:

$$y_q^{(M)} = \bigvee_{i=0}^{M-1} \zeta_{q,i}, \tag{4.6}$$

corresponding to the N combiners each made of an M-input logic-or in the bottom part of Figure 4.11.Pipelining the proposed architecture improves the throughput, but leads to an area overhead. As an example, adding one pipeline register between each of the W stages in Figure 4.12, (i.e. $W - 1$ pipeline registers), implies adding $W - 1 - q$ registers to each $y_q^{(M)}$, to increase the throughput by about W times.

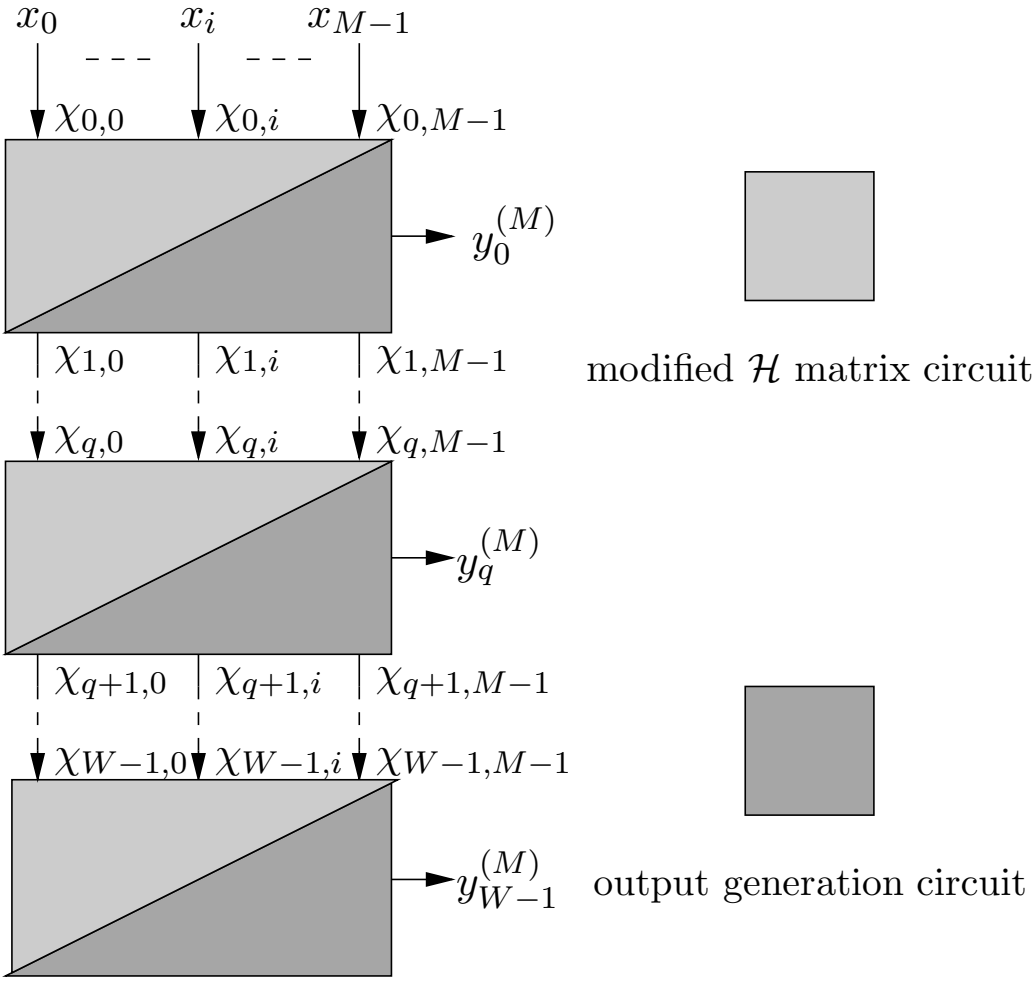


Figure 4.12: BWA architecture: cascade of W stages.

4.3 Comparator-based VLSI Architectures with High Speed[2]

In last section a BWA architecture based on Radix-Sort with very low hardware cost is introduced, here we are about to present two other architectures based on comparators of high parallelization for finding the first W maximum/minimum values from M inputs, which achieves a quite good timing performance. Being Similar with that in Section 4.2.1, we redefine the problem of finding the first W maximum/minimum values as follows for the sake of convenience. Given a set $\mathcal{X}^{(M)} = \{x_0, \ldots, x_{M-1}\}$ made of M elements we want to find the first W maximum/minimum values, namely $\mathbf{y}^{(M)} =$

[2]Content of this section is partly from the paper: Guoing XIAO, Waqar AHMAD, Syed Azhar Ali ZAIDI, Massimo RUOROCH, CAUSAPRUNO, "High Speed VLSI Architecture for Finding the First W Maximum/minimum Values", International Conference on Electronics Application, Rome, May 2014

$\{y_0^{(M)}, y_1^{(M)}, \ldots, y_z^{(M)}, \ldots, y_{W-1}^{(M)}\}$ where $y_0^{(M)} = \max(\mathcal{X}^{(M)})$, $y_1^{(M)} = \max(\mathcal{X}^{(M)} \setminus \{y_0^{(M)}\})$, $\ldots$, $y_z^{(M)} = \max(\mathcal{X}^{(M)} \setminus \bigcup_{k=0}^{z-1}\{y_k^{(M)}\})$, $\ldots$, $y_{W-1}^{(M)} = \max(\mathcal{X}^{(M)} \setminus \bigcup_{k=0}^{W-2}\{y_k^{(M)}\})$ (similarly substituting max with min). For the sake of simplicity in the following we will discuss only the max case as the min equivalent solution can be straightforwardly derived.

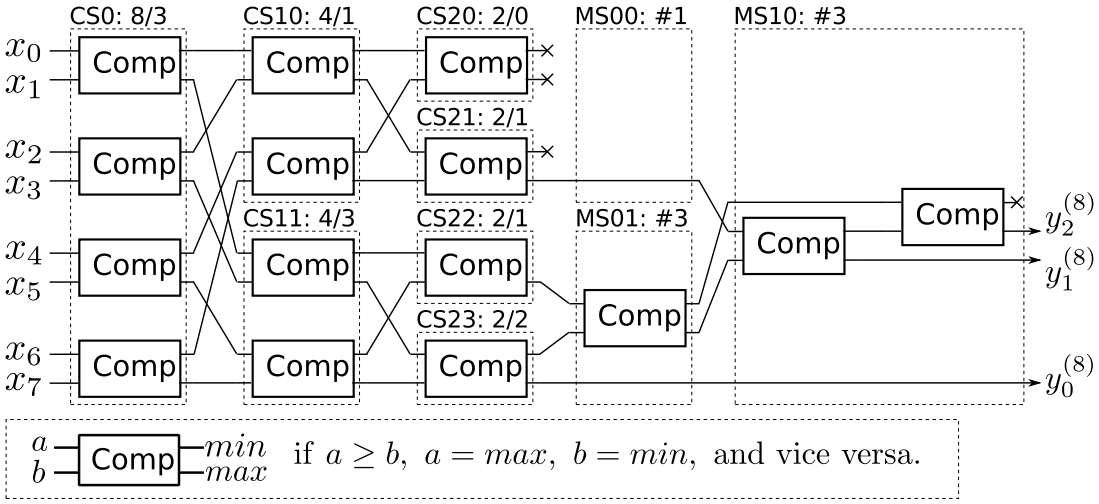


Figure 4.13: An example: Structure for finding the first 3 maximum values from 8 inputs

4.3.1 The Architecture of Partial Sorting (PS)

Algorithm of Partial Sorting

Tree-like architecture using comparators is the most intuitive way to obtain maximum or minimum values, which is utilized in [89] together with some control logics for finding the first two minimums. While in [95] an optimized sorting network derived from tree-like structure for extracting the first 3 minima from 32 inputs is presented. In this work we extend this kind of architecture to a general case of $w > 2$ as detailed in the following.

As known to all, it is easy to get the maximum value using the tree-like structure after several comparing stage, but for the other maxima some extra work needs to be done. More specifically, we take the case of $M = 8$ and $W = 3$, where the input set is $\mathcal{X}^{(8)} = \{x_0, x_1, \ldots, x_7\}$ and the output is $\mathbf{y}^{(8)} = \{y_0^{(8)}, y_1^{(8)}, y_2^{(8)}\}$. As depicted in Figure 4.13, there are three comparing stages, CS0, CS1 and CS2, which are followed by two stages of merging operation, MS0 and MS1. The first block of CS1 (labeled as CS01) which takes the 4 relative minimum values as input may contains one of the first three maximum values, while the second block, CS11, possibly includes all the 3 target values. The similar analysis can also be applied in the third comparing stage, in which CS21, CS22 and CS23 probably own 1, 1 and 2 of the three maximum values respectively, but CS20 definitely contains no maximum candidate which can be removed indeed. After all

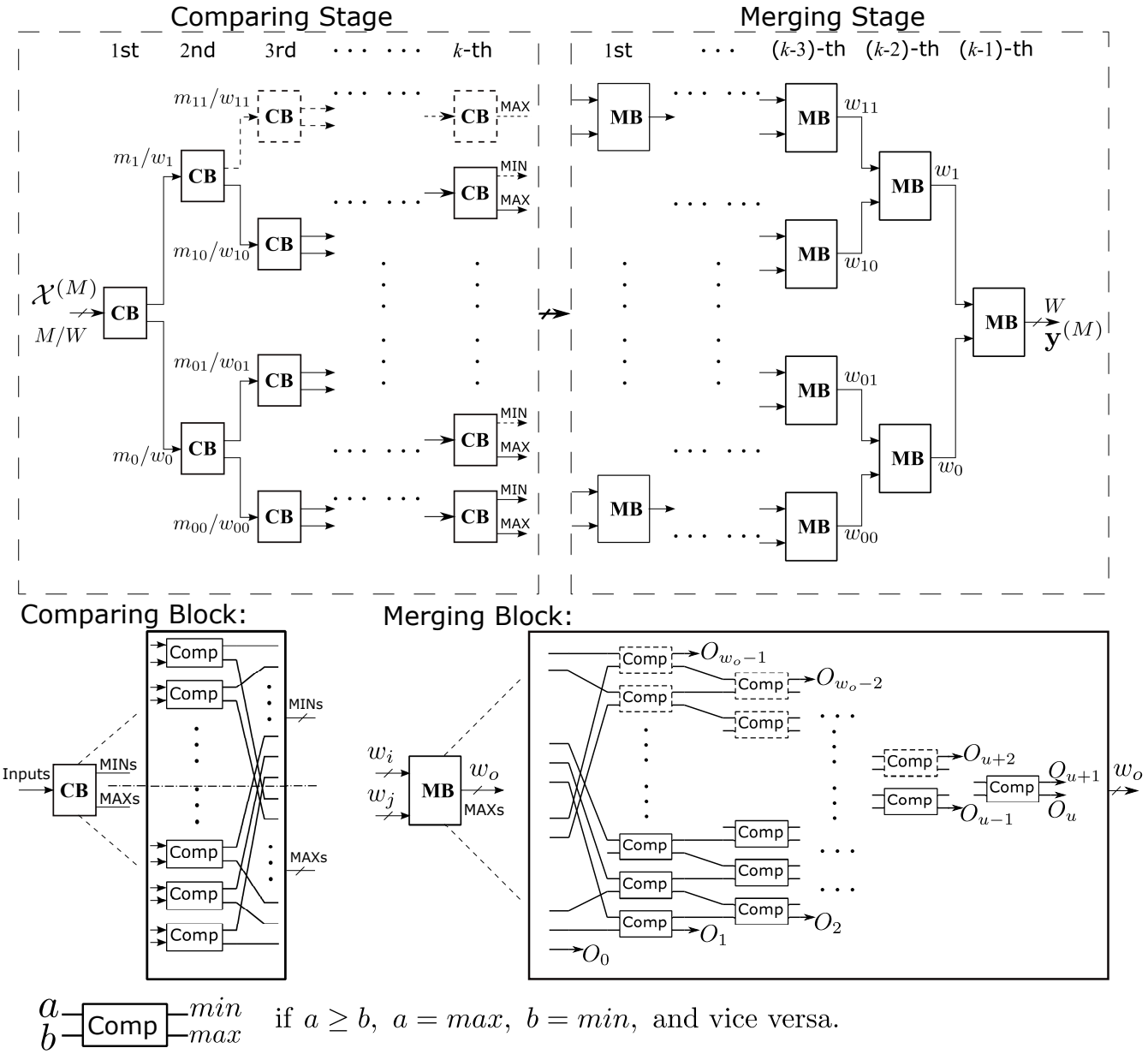


Figure 4.14: Structure of Partial-sorting for finding W maximum values from M inputs

the possible maximum candidates are obtained by the comparing network, the following merging operations will find the target values from them. We can see from the figure that one comparator is used in MS01 which may hold 3 maximum candidates to sort the inputs from CS22 and CS23 into a descending order, meantime MS00 needs to do nothing since it has only one input. Then the MS10 takes the outputs from MS00 and MS01 and performs a similar operation to find the first 3 maximum values finally. Since in the comparing stage some of the branches of the network could be cut off like CS20 in the previous example we name this algorithm as *Partial Sorting*.

Architecture implementation

In general, the general PS architecture for finding the first W maximum values from M inputs is more complicated than that depicted in Figure 4.13, which consists of two sequential stages: *Comparing stage* and *Merging stage* shown in Figure 4.14. In the first step of Comparing stage, M inputs are divided into two groups after performing a comparison between every two adjacent values, where the larger value goes into the upper group, called MINs-group, while the smaller one stays in the lower group (MAXs-group). As a result we can easily figure out that there are at most $W/2$ maximum values among the first W maximum values which may be contained in the MINs-group of the first round, and similarly the MAXs-group possibly includes all the W target values. Considering the case that M and W would be odd numbers, the m_0, w_0, m_1 and w_1 are

$$\begin{cases} m_0 = m_1 = \dfrac{M}{2} & \text{if } M \text{ is even} \\ m_0 = [\dfrac{M}{2}] + 1,\ m_1 = [\dfrac{M}{2}] & \text{if } M \text{ is odd} \\ w_0 = W, w_1 = [\dfrac{W}{2}] & \end{cases} . \tag{4.7}$$

where $W \leq M/2$ is initially defined and the operator $[\bullet]$ is to take the integer part of the operating value. As shown in Figure 4.14, in the zoomed structure of Comparing Block we represent the last input (top-down) with a dash line to indicate that when the number of inputs is odd, the alone input not involved in comparison of current round is arranged into the MAXs-group directly. In a similar way, the same operation is applied on the values of their MINs-groups and MAXs-groups respectively in the next round, in which the corresponding parameters are obtained through

$$\begin{cases} m_{00} = m_{01} = \dfrac{m_0}{2} & \text{if } m_0 \text{ is even} \\ m_{00} = [\dfrac{m_0}{2}] + 1,\ m_{01} = [\dfrac{m_0}{2}] & \text{if } m_0 \text{ is odd} \\ w_{00} = w_0, w_{01} = [\dfrac{w_0}{2}] & \text{if} w_0 \leq m_{00} \\ w_{00} = m_{00},\ w_{01} = [\dfrac{w_0}{2}] & \text{if} w_0 > m_{00} \end{cases} . \tag{4.8}$$

$$\begin{cases} m_{10} = m_{11} = \dfrac{m_1}{2} & \text{if } m_1 \text{ is even} \\ m_{10} = [\dfrac{m_1}{2}] + 1,\ m_{11} = [\dfrac{m_1}{2}] & \text{if } m_1 \text{ is odd} \\ w_{10} = w_1, w_{01} = [\dfrac{w_1}{2}] & \text{if} w_1 \leq m_{10} \\ w_{10} = m_{10},\ w_{01} = [\dfrac{w_1}{2}] & \text{if} w_1 > m_{10} \end{cases} . \tag{4.9}$$

This procedure is repeated iteratively until to the k-th round, where $k = [log_2(M -$

1)] + 1. Generally the parameters, the size of sub-groups (including MINs- and MAXs-) and the number of possible maximum value contained in each sub-group, can be derived from their own parents' group in similar way with 4.8 and 4.9, for example in the k-th round $w_{0...00}$ and $w_{0...01}$ can be derived from $w_{0...0}$ like above. Specially in upper part of Figure 4.14, the dash boxes and the dash arrows within comparing stage mean that they would not be needed when the number of possible maximum value reaches one, since in that case no maximum value will fall into its sub-branch of MINs-group and it will be cut off.

After the Comparing stage generates all possible coarse candidates, the Merging Stage will merge them in an inverse way to Comparing Stage and finally produce the first W maximum values. The Merging Block's main task is to sort and merge the results generated by the two previous groups of Comparing Stage in a decreasing order. The details of the Merging Block (MB) is depicted in the bottom-right part of Figure 4.14.

It is obvious that only one kind of component, namely comparator, is applied in the whole architecture of Partial sorting and no control logic is utilized. The hardware complexity and the latency of the structure can be derived by $O(M\log_2(M))$ and $O(W\log_2(M))$ respectively. As in this algorithm the comparing and merging network is derived in a dichotomously and heuristically way. Although it is not feasible to give a determined expression for a general case on the hardware cost and delay, the experimental results show that it achieves an excellent timing performance at an acceptable cost of hardware.

4.3.2 The Architecture using Fully Parallellized Comparision Grid (FPCG)

Algorithm Description

The work in [89, 90] has presented novel comparator-based architectures for finding the first $W = 2$ maximum/minimum values, which rely on N levels of comparing stages with each level l of O_l comparing modules. These structures can be extended to $W > 2$ as illustrated in Figure 4.15 and detailed in the following. The i-th comparing stage of level $l = 1$ compares $K_{1,i}$ elements taken from $\mathcal{X}^{(M)}$ and produces W results, leading to $K_{1,i} \geq W$. In the following we will refer to $K_{l,i}$ as the radix of the comparing stage, where

$$M = \prod_{l=1}^{N} \dot{K}_l \qquad \dot{K}_l = \frac{1}{O_l} \sum_{i=0}^{O_l-1} K_{l,i}. \tag{4.10}$$

Let $\mathcal{X}^{(K_{1,i})} \subseteq \mathcal{X}^{(M)}$ be the set of elements compared by the i-th stage of level $l = 1$, with $\bigcup_{i=0}^{O_1-1} \mathcal{X}^{(K_{1,i})} = \mathcal{X}^{(M)}$, $\mathcal{X}^{(K_{1,i})} \bigcap \mathcal{X}^{(K_{1,j})} = \emptyset$ and $i, j = 0, \ldots, O_1 - 1$, $i \neq j$. Similarly, we

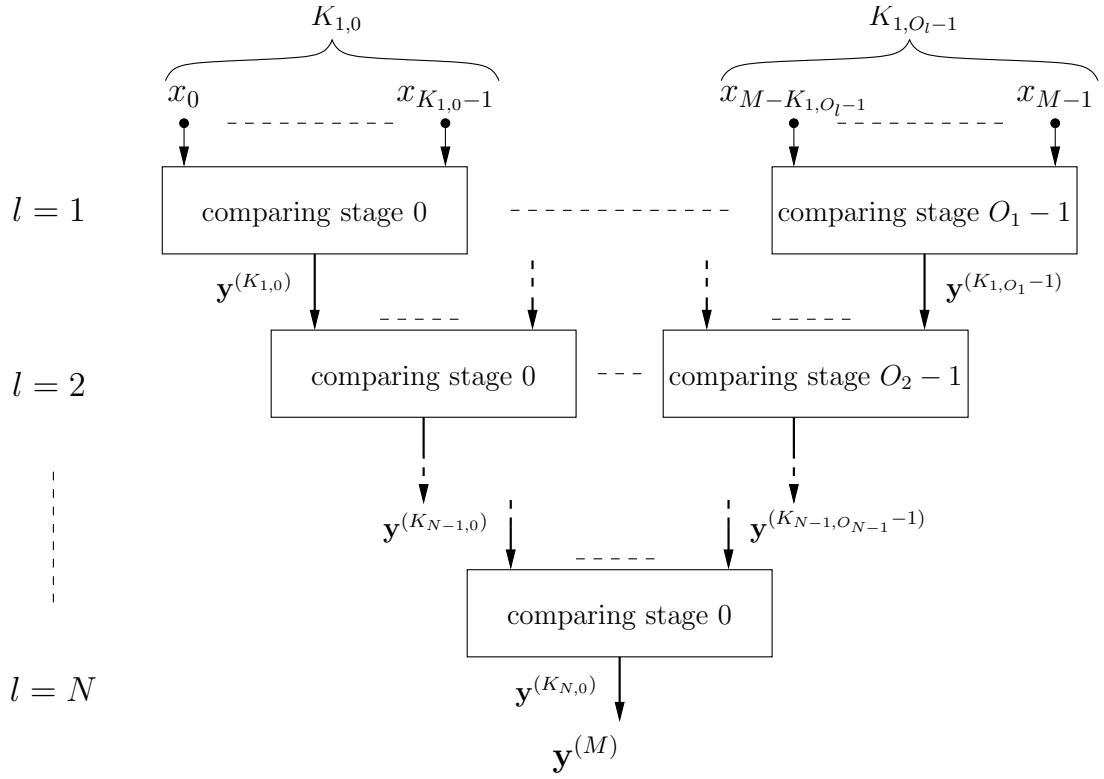


Figure 4.15: Tree structure for finding the first W maximum values.

define $y_0^{(K_{1,i})} = \max(\mathcal{X}^{(K_{1,i})})$ and $y_z^{(K_{1,i})} = \max(\mathcal{X}^{(K_{1,i})} \backslash \bigcup_{k=0}^{z-1}\{y_k^{(M)}\})$ with $z = 1, \ldots, W-1$ as the output of the i-th stage of level $l = 1$. The number of comparators required by a comparing stage depends on its radix and the total number of comparators at $l = 1$ is

$$C_{l=1} = \sum_{i=0}^{O_1-1} \left(\begin{array}{c} K_{1,i} \\ 2 \end{array} \right) = \sum_{i=0}^{O_1-1} \frac{K_{1,i} \cdot (K_{1,i} - 1)}{2}. \tag{4.11}$$

If all the comparing stages have the same radix, (4.11) simplifies to $M \cdot (K_1 - 1)/2$, where K_1 is the radix at $l = 1$.

A similar approach can be followed for $l > 1$. In this case, the i-th comparing stage receives from the previous level $K_{l,i}$ arrays, where the j-th array from level $l-1$ contains W sorted values, namely $\mathbf{y}^{(K_{l-1,j})} = y_0^{(K_{l-1,j})}, \ldots, y_{W-1}^{(K_{l-1,j})}$. It is worth noting that $K_{l,i} \geq 2$ for $l > 1$ and the only constraint is (4.10). The number of comparators required by each comparing stage is $\alpha_W \cdot \beta_{K_{l,i}}$, where α_W is the number of comparators to compute the first W maximum values from two arrays made of W sorted values and $\beta_{K_{l,i}}$ is the number of couples of arrays (without repetition) we can compose from $K_{l,i}$ arrays. Thus, $\beta_{K_{l,i}} = K_{l,i} \cdot (K_{l,i} - 1)/2$ whereas α_W can be obtained as follows. Let $\mathbf{y}' = y'_0, \ldots, y'_{W-1}$ and $\mathbf{y}'' = y''_0, \ldots, y''_{W-1}$ be two sorted arrays of W elements. Then, we observe that $y_0 = \max(\mathbf{y}', \mathbf{y}'') = \max(y'_0, y''_0)$ requires one comparator ($\alpha_1 = 1$). Finding $y_1 = \max(\{\mathbf{y}', \mathbf{y}''\} \backslash$

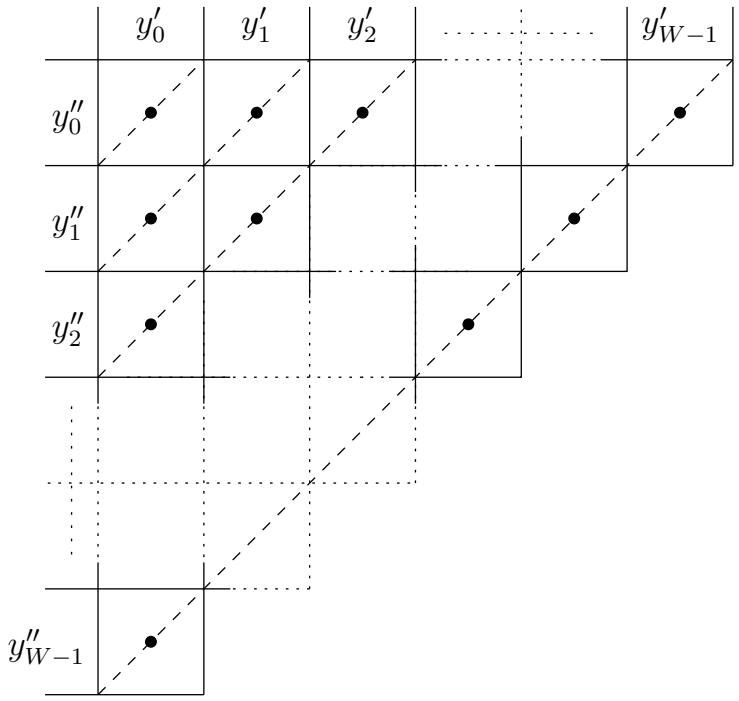


Figure 4.16: Graphical representation of the number of comparators required to find the first W maximum values in $\mathbf{y}'$ and $\mathbf{y}''$

$\{y_0\}$) implies to compare y'_0 with y''_0, y'_0 with y''_1 and y''_0 with y'_1 ($\alpha_2 = 3$). The first comparison has already been performed to find y_0 and so two further comparators are required to obtain

$$y_1 = \begin{cases} y''_0 & \text{if } y'_0 \geq y''_0 \text{ and } y''_0 \geq y'_1 \\ y'_1 & \text{if } y'_0 \geq y''_0 \text{ and } y''_0 < y'_1 \\ y'_0 & \text{if } y'_0 < y''_0 \text{ and } y'_0 \geq y''_1 \\ y''_1 & \text{otherwise} \end{cases} . \tag{4.12}$$

This analysis can be extended to $y_z = \max(\{\mathbf{y}', \mathbf{y}''\} \setminus \bigcup_{k=0}^{z-1}\{y_k\})$ by using a graphical representation of the problem. As shown in Figure 4.16 we build a grid where the elements of $\mathbf{y}'$ and $\mathbf{y}''$ are the vertical and horizontal labels respectively. Then, we can infer that $\alpha_z = \alpha_{z-1} + z$, namely the number of comparators to find the first z maximum values, are the dots contained in the triangle delimited by the diagonal of the $z \times z$ square on the grid. Thus, $\alpha_W = \sum_{k=1}^{W} k = W \cdot (W+1)/2$ and the total number of comparators is

$$C = C_{l=1} + C_{l>1} = \sum_{l=1}^{N} \sum_{i=0}^{O_l - 1} \gamma_l \cdot \frac{K_{l,i} \cdot (K_{l,i} - 1)}{2} \tag{4.13}$$

where

$$\gamma_l = \begin{cases} 1 & \text{if } l = 1 \\ \dfrac{W \cdot (W+1)}{2} & \text{otherwise} \end{cases}. \tag{4.14}$$

If all the comparing stages within a level have the same radix ($K_{l,i} = K_l$), then (4.13) can be rewritten as

$$C = \frac{M \cdot (K_1 - 1)}{2} + \frac{W \cdot (W+1) \cdot M}{4} \cdot \sum_{l=2}^{N} \frac{K_l - 1}{\prod_{r=1}^{l-1} K_r}. \tag{4.15}$$

Moreover, if all the levels have the same radix ($K_l = K$), then (4.15) simplifies to

$$C = \frac{M \cdot (K-1)}{2} + \frac{W \cdot (W+1)}{2} \cdot \frac{M-K}{2}. \tag{4.16}$$

In the following an architecture with $K_{l,i} = K_l = K$ will be referred to as constant-radix architecture. As shown, in [90], finding the optimal set of radix to minimize the number of comparators in (4.13) and (4.15) is a Diophantine problem with integer coefficients that not always admits solutions in $\mathbb{N}$. On the other hand, an interesting result can be obtained from (4.16) by computing the partial derivative of C with respect to K:

$$\xi = \frac{\partial C}{\partial K} = \frac{M - 2W - 2W^2}{4}. \tag{4.17}$$

Indeed, ξ is a parabolic function of W. Thus, given M and W we can understand if the slope of C is positive or negative:

$$\begin{array}{l} \xi > 0 \quad \text{if} \quad W < W^* \\ \xi < 0 \quad \text{if} \quad W > W^* \end{array} \qquad W^* = \frac{1}{2}\left(1 + \sqrt{1 + 8M}\right) \tag{4.18}$$

Thus, if the slope is positive (4.16) is minimized by $K = W$. On the contrary, if the slope is negative (4.16) is minimized by $K = M$ (maximum radix architecture with a one-level tree). However, since the second term in (4.15) grows as W^2, the solution obtained with $K = W$ is rather large: $C_{K=W} = M \cdot (W-1)/2 + W \cdot (W+1) \cdot (M-W)/4$. Thus, we relax the constant-radix constraint to $K_1 = \check{K}$ and $K_{l>1} = \mathring{K}$. In this case (4.15) becomes

$$C = \frac{M \cdot (\check{K} - 1)}{2} + \frac{W \cdot (W+1) \cdot \mathring{K}}{4} \cdot \left(\frac{M}{\check{K}} - 1\right). \tag{4.19}$$

Then, computing the partial derivative of C with respect to $\check{K}$ and $\mathring{K}$ we obtain

$$\xi_1 = \frac{\partial C}{\partial \check{K}} = \frac{M}{2} - \frac{W \cdot (W+1) \cdot M \cdot \mathring{K}}{4(\check{K})^2} \tag{4.20}$$

$$\xi_2 = \frac{\partial C}{\partial \mathring{K}} = \frac{W \cdot (W+1)}{4} \left(\frac{M}{\check{K}} - 1 \right). \tag{4.21}$$

Since ξ_2 is always positive, (4.19) is minimized by $\mathring{K} = 2$. Moreover, with $\mathring{K} = 2$ we obtain that $\xi_1 \approx 0$ for $\check{K} \approx W$ and so $C_{K_1=W,K_{l>1}=2} = M \cdot (W-1)/2 + (W+1) \cdot (M-W)/2$. Since $W > 2$ we obtain that $C_{K_1=W,K_{l>1}=2} < C_{K=W}$.

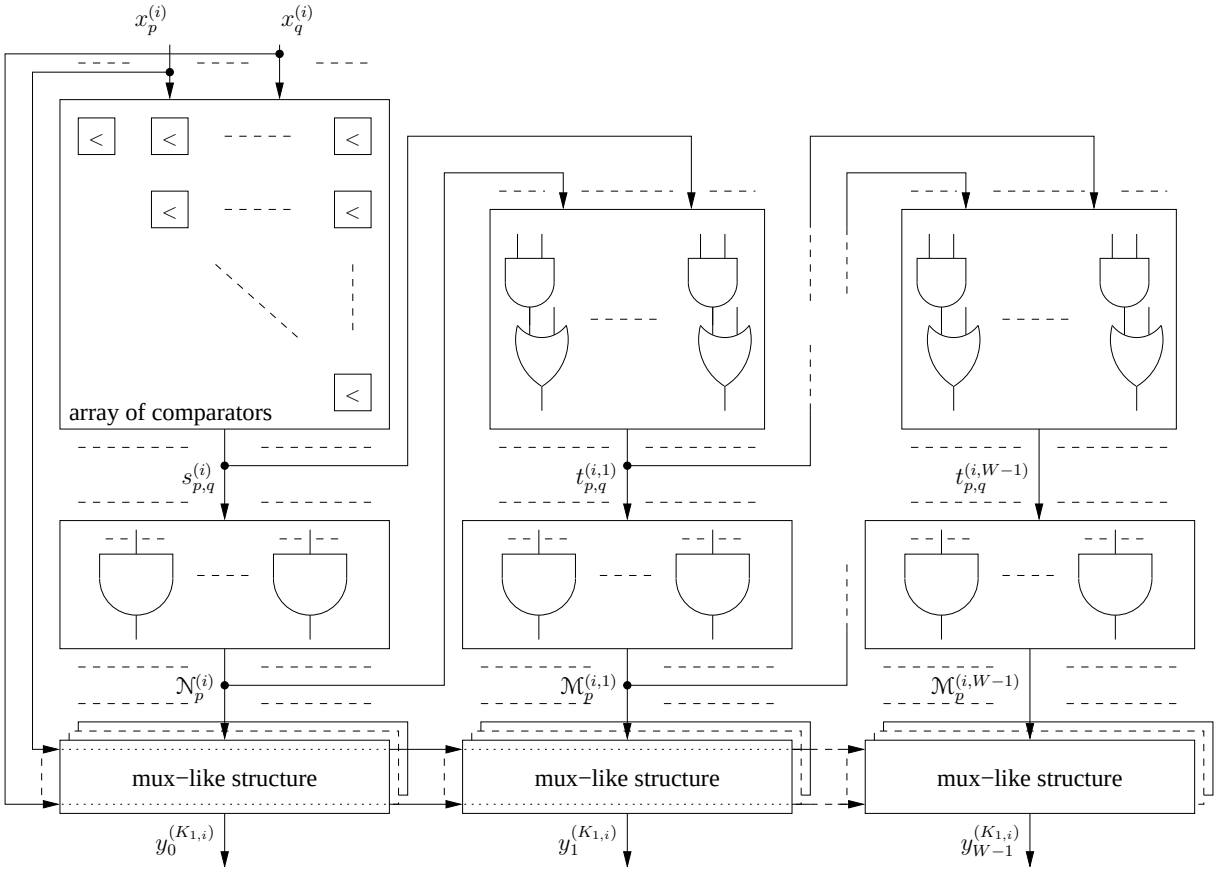


Figure 4.17: Architecture of a comparing stage at $l = 1$

Architecture implementation

As highlighted in Section 4.3.2, for each M and W values, mainly two architectures worth investigation: i) $K = M$ and ii) $K_1 = W$, $K_{l>1} = 2$. Moreover, the number of comparators required by comparing stages at $l = 1$ differs from the one at $l > 1$. This difference reflects in different architectures in the two cases. Thus, for the sake of clarity, in the following the two architectures are detailed separately.

(1) $\mathbf{l = 1}$ **comparing stages**: Both $K_1 = W$ and $K = M$ architectures require at least one level of comparing stages. Thus, at $l = 1$ a radix-$K_{1,i}$ comparing stage computes the first W maximum values among its $K_{1,i}$ inputs by the means of $K_{1,i} \cdot (K_{1,i} - 1)/2$ comparators working concurrently. Let $x_p^{(i)}, x_q^{(i)} \in \mathcal{X}^{K_{1,i}}$ be two inputs of the i-th comparing stage. Let $s_{p,q}^{(i)}$ be the result of the comparison between $x_p^{(i)}$ and $x_q^{(i)}$, namely $s_{p,q}^{(i)}$ is the sign of $x_p^{(i)} - x_q^{(i)}$:

$$s_{p,q}^{(i)} = \begin{cases} 0 & \text{if } x_p^{(i)} > x_q^{(i)} \\ 1 & \text{if } x_p^{(i)} < x_q^{(i)} \\ - & \text{if } x_p^{(i)} = x_q^{(i)} \end{cases} \tag{4.22}$$

where $-$ represents a don't-care. If $x_n^{(i)}$ is the first maximum, then $s_{q,n}^{(i)} = 1$ for every q such that $0 \leq q \leq K_{1,i} - 1$ and $q \neq n$. Let $\mathcal{N}^{(i)}$ be the array whose p-th element is

$$\mathcal{N}_p^{(i)} = \bigwedge_{q=0, q\neq p}^{K_{1,i}-1} s_{q,p}^{(i)} \tag{4.23}$$

where $\bigwedge$ represents the logic-and operation. As it can be observed, $\mathcal{N}^{(i)}$ is the one-hot representation of n and can be used as the selection signal of a mux-like structure. The mux-like structure is the same one proposed in [90], namely for each $x_u^{(i)} \in \mathcal{X}^{(K_{1,i})}$ with $0 \leq u \leq K_{1,i} - 1$ let $x_{u,v}^{(i)}$ be the v-th bit of $x_u^{(i)}$ and d the number of bits to represent $x_u^{(i)}$. Then, $y_{0,v}^{(K_{1,i})}$, the v-th bit of $y_0^{(K_{1,i})}$ is

$$y_{0,v}^{(K_{1,i})} = \bigvee_{u=0}^{K_{1,i}-1} x_{u,v}^{(i)} \wedge \mathcal{N}_u^{(i)} \tag{4.24}$$

where $\bigvee$ is the logic-or operation.

The approach suggested in [90] to find $y_1^{(K_{1,i})}$ by the means of a masking circuit can be extended in this work to find $y_n^{(K_{1,i})}$ with $n = 1, \ldots, W-1$ (see Figure4.17). As argued in [90] the following formulation should ease understanding the underlying idea, even if, from a formal point of view, it has some redundancy. Let $\hat{t}_{p,q}^{(i)} = s_{p,q}^{(i)} \wedge \overline{\mathcal{N}_q^{(i)}}$ where $\overline{(\cdot)}$ is the logic-not operation and $\hat{t}_{p,q}^{(i)} = 1$ if $x_q^{(i)} > x_p^{(i)}$ and $x_q^{(i)} \neq y_0^{K_{1,i}}$. Then, let $\mathcal{M}^{(i)}$ be the array whose p-th element is

$$\mathcal{M}_p^{(i)} = \bigwedge_{q=0, q\neq p}^{K_{1,i}-1} t_{q,p}^{(i)} \tag{4.25}$$

where $t_{q,p}^{(i)} = \hat{t}_{q,p}^{(i)} \vee \mathcal{N}_q^{(i)}$. If x_m is the second maximum value, then $\mathcal{M}^{(i)}$ is the one-hot representation of m and it can be used as the selection signal of a mux-like structure to

obtain $y_1^{(K_{1,i})}$:

$$y_{1,v}^{(K_{1,i})} = \bigvee_{u=0}^{K_{1,i}-1} x_{u,v}^{(i)} \wedge \mathcal{M}_u^{(i)} \tag{4.26}$$

as shown in Figs. 4.17 and 4.18. Thus, $y_z^{(K_{1,i})}$ can be obtained as

$$y_{z,v}^{(K_{1,i})} = \bigvee_{u=0}^{K_{1,i}-1} x_{u,v}^{(i)} \wedge \mathcal{M}_u^{(i,z)} \tag{4.27}$$

where

$$\mathcal{M}_u^{(i,z)} = \bigwedge_{q=0,q\neq u}^{K_{1,i}-1} t_{q,u}^{(i,z)} \tag{4.28}$$

and

$$t_{q,u}^{(i,z)} = \left(t_{q,u}^{(i,z-1)} \wedge \overline{\mathcal{M}_u^{(i,z-1)}} \right) \vee \mathcal{M}_q^{(i,z-1)} \tag{4.29}$$

with $t_{q,u}^{(i,0)} = s_{q,u}^{(i)}$ and $\mathcal{M}_q^{(i,0)} = \mathcal{N}_q^{(i)}$.

(2) **$\mathbf{l > 1}$ comparing stages**: As discussed in Section 4.3.2 for comparing stages at $l > 1$ the case $K_{l>1} = 2$ worths investigation. Thus, this section concentrates on comparing stages receiving two sorted arrays each of which contains W elements. Since all the comparing stages have the same structure, in the following we will refer to the two sorted arrays input to the i-th comparing stage at $l > 1$ as $\mathbf{y}'$ and $\mathbf{y}''$ for the sake of simplicity. Then, given that $\mathbf{y} = \{y_0, y_1, \ldots, y_{W-1}\}$ is the array containing the first W maximum values taken from $\mathbf{y}'$, $\mathbf{y}''$ and $\mathcal{Y}_n = \{y'_n, y''_n\}$ we have that $y_n \in \mathcal{Y}|_0^n$, where $\mathcal{Y}|_0^n = \bigcup_{r=0}^n \mathcal{Y}_r$. As a consequence, the architecture to find y_n relies on multiplexers that, based on the output of the comparators, select the correct result. The logic circuit that combines the output of the comparators to generate the signals to drive the multiplexers can be derived as described in the following. The possible values of $\mathbf{y}$ can be represented as a tree and, if y_{n-1} is known, then y_n belongs to a subset of $\mathcal{Y}|_0^n$. As an example $y_1 \in \mathcal{Y}_0^1 = \{y'_0, y''_0, y'_1, y''_1\}$, but if $y_0 = y''_0$ then $y_1 \in \{y'_0, y''_1\}$. Thus, if f_n represents the choice made at level n in the tree to choose y_n (being known y_{n-1}) then the multiplexers that select y_n are driven by a combination of f_0, f_1, …, f_{n-1}. Due to the simmetry of the problem, we can simplify the tree representation introducing y_n^a, where

$$a = \begin{cases} 0 & \text{if } y_n^a = y'_n \\ 1 & \text{if } y_n^a = y''_n \end{cases} . \tag{4.30}$$

Thus, if $y_{n-1} = y_i^a$ with $0 \leq i \leq n-1$, then $y_n \in \{y_{i+1}^a, y^{\overline{a}}{}_j\} \subseteq \mathcal{Y}_0^n$ with $j \leq i$. It is worth noting that once y_{n-1} is known, then y_n belongs to a subset of $\mathcal{Y}_0^n$ that contains only two elements. As a consequence, if the sequence $f_0, \ldots, f_{n-1}$ is known, then f_n is a binary

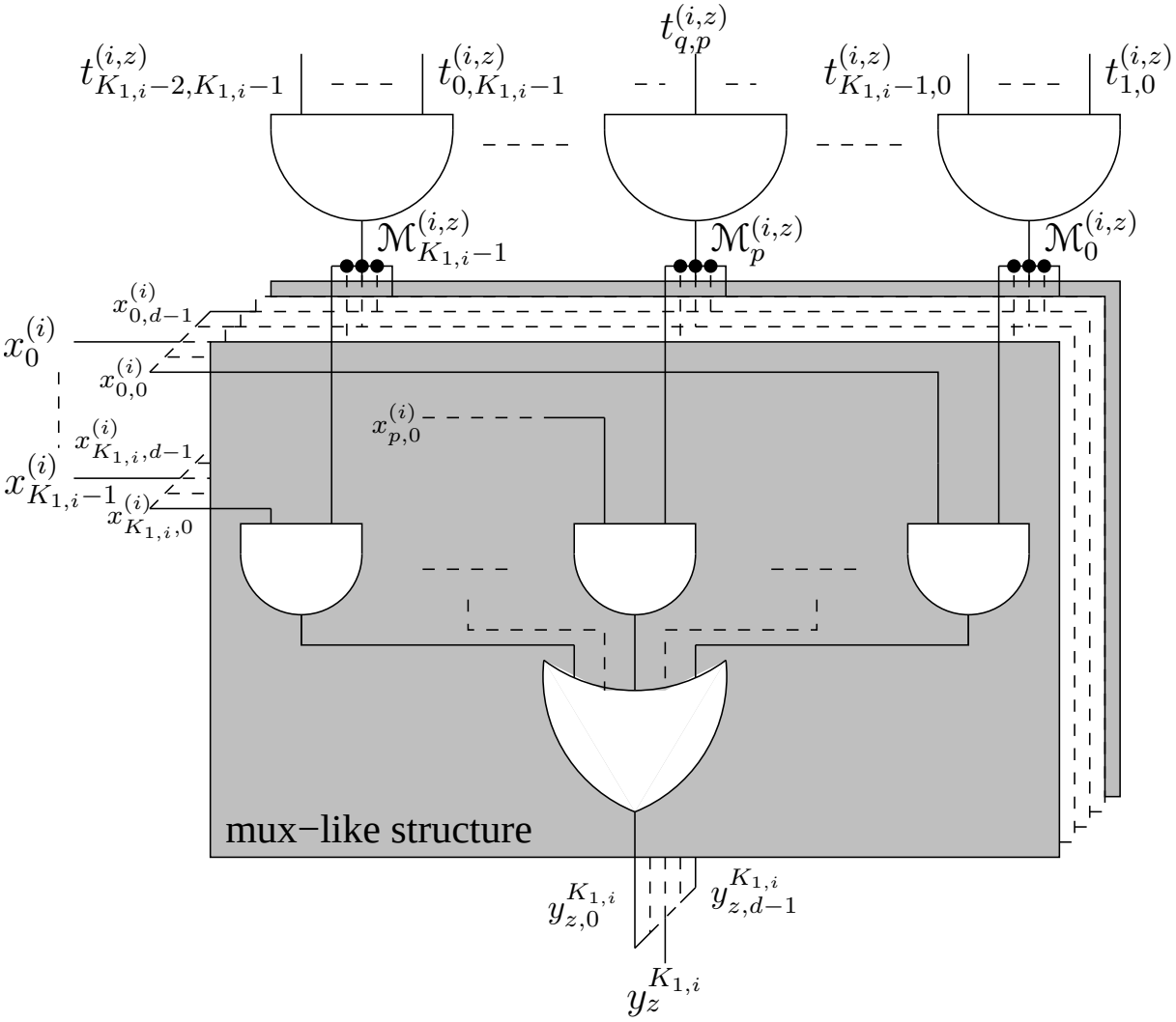


Figure 4.18: Details of the architecture of a comparing stage at $l = 1$: one-hot signals and mux-like structure

value that can be represented on a tree as well. Let us define $f_n^{i,j}$ as the value taken by f_n when comparing y_i^a and $y_j^{\overline{a}}$, namely if $i < j$ then $f_n^{i,j} = \text{sgn}(y_i^a - y_j^{\overline{a}})$ where $\text{sgn}(\cdot)$ is the sign function. On the other hand, if $i = j$ then we assume

$$f_n^{i,i} = \begin{cases} a & \text{if } y_i^a \geq y_i^{\overline{a}} \\ \overline{a} & \text{otherwise} \end{cases} . \tag{4.31}$$

Thus, in a tree representation the possible $\mathbf{y}$ sequences are the paths from the root to the leaves whereas nodes represent $f_n^{i,j}$, Figure 4.19 shows the tree obtained for $W = 5$. The logic circuit to select y_n is derived exploring the tree. Depending on the exploration strategy the multiplexer sequence to select y_n changes and the selection circuit changes accordingly. In Figure 4.20 a possible circuit for $W = 5$ is shown. As it can be observed,

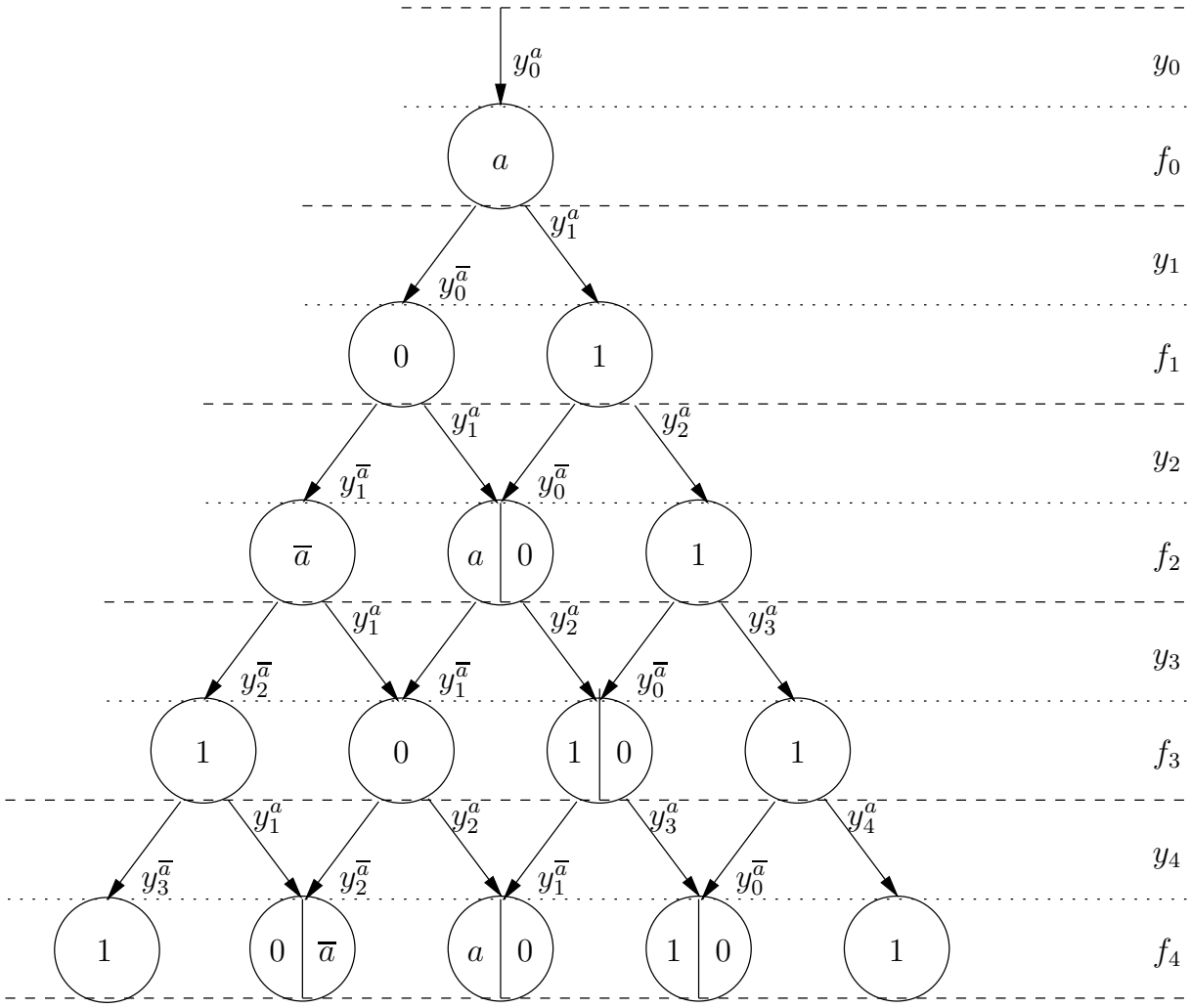


Figure 4.19: Tree representation of the possible $\mathbf{y}$ sequences and f_n values for $W = 5$.

the selection circuit for y_0 and y_1 is the same one proposed in [89] for $W = 2$. Besides, the right-most path of the tree in Figure 4.19 is characterized by $f_n = 1$ for $n = 1, \ldots, W-1$. As a consequence, $y_n \in \{y_0^{\overline{a}}, y_n^a\}$ is selected by $\bigwedge_{i=1}^{n-1} f_i$, so $y_2 \in \{y_0^{\overline{a}}, y_2^a\}$, $y_3 \in \{y_0^{\overline{a}}, y_3^a\}$ and $y_4 \in \{y_0^{\overline{a}}, y_4^a\}$ are selected by $\sigma_2^{0,2} = f_1$, $\sigma_3^{0,3} = f_2 \wedge f_1 = f_2 \wedge \sigma_2^{0,2}$ and $\sigma_4^{0,4} = f_3 \wedge f_2 \wedge f_1 = f_3 \wedge \sigma_3^{0,3}$ respectively. Thus, $\sigma_n^{0,n} = f_{n-1} \wedge \sigma_{n-1}^{0,n-1}$ with $\sigma_2^{0,2} = f_1$. Moreover, the left-most path of the tree in Figure 4.19 shows that $\sigma_3^{1,2} = \overline{f_1} \wedge \overline{a} = \overline{f_1} \wedge \overline{f_0}$ and $\sigma_4^{1,3} = f_3 \wedge \overline{f_1} \wedge \overline{f_0} = f_3 \wedge \sigma_3^{1,2}$ select $y_3 \in \{y_1^a, y_2^{\overline{a}}\}$ and $y_4 \in \{y_1^a, y_3^{\overline{a}}\}$. Furthermore, Figure 4.19 shows that $y_4 \in \{y_2^a, y_2^{\overline{a}}\}$ is reached by three possible paths i) $\overline{f_3} \wedge \overline{a} \wedge \overline{f_1}$, ii) $\overline{f_3} \wedge a \wedge \overline{f_1}$, iii) $\overline{f_3} \wedge \overline{f_2} \wedge f_1$, that correspond to $\sigma_4^{2,2} = \overline{f_3} \wedge (\overline{f_2} \vee \overline{f_1})$.

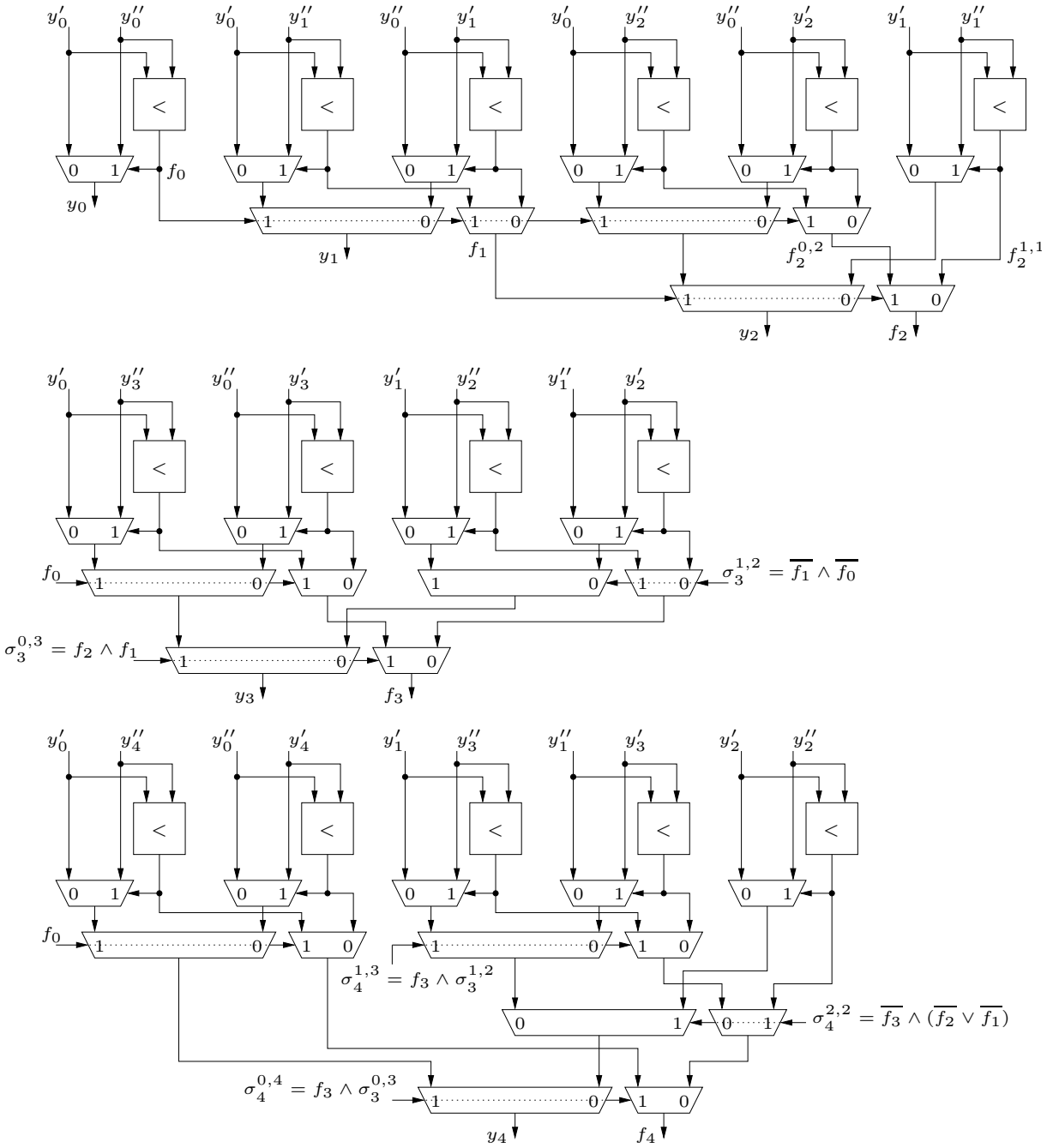


Figure 4.20: Circuit derived from the tree representation in Figure 4.19.

4.4 Experimental Results and Comparisons

Experimental results obtained in the context of i) K-best MIMO detectors, ii) Non-binary LDPC decoder architectures and iii) Turbo product code architectures are shown and compared with solutions presented in the literature. Since several works do not give complete synthesis results, we re-implement the solutions presented in [91, 93, 95, 96] as

stand-alone units. Moreover, we include the Partial Bitonic (PB) architecture proposed in [97]. Finally, the SN derived from [75] and proposed in [74] is summarized in the following paragraphs and included in the comparison as well.

SN review

The SN described in [74] is a special case of sorting network that moves the W largest values out of $M = 2W$ inputs to the first W output lines ($2W/W$ SN). It relies on two W-element sorters and a $2W$-element pruned-merger, depicted as two solid-line boxes and one dashed-line box respectively in Figure 4.21 (a). In this work the sorters are implemented as even-odd Butcher sorting networks [74] and the pruned-merger is made of W comparators to select the W largest values. As argued by [74] this network can be extended to the case $M = n \cdot W$ by using $n - 1$ instances of the $2W/W$ SN. Unfortunately, the W maximum values obtained with this solution are not sorted. Thus, for a fair comparison with the proposed BWA architecture the SN is connected to a further W-element sorter. The general block scheme of the SN is shown in Figure 4.21 (b).

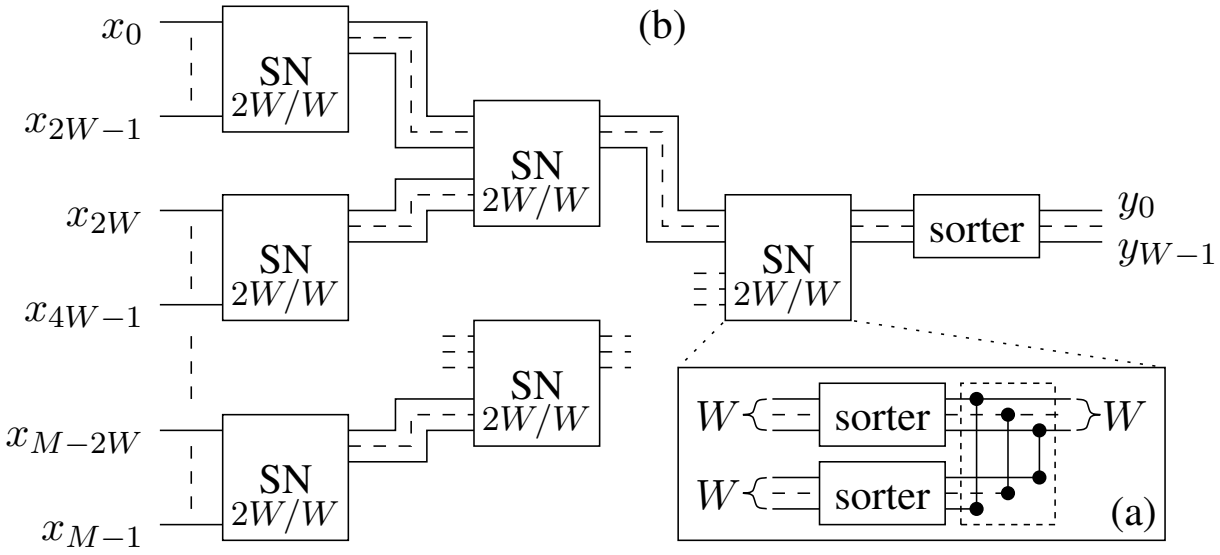


Figure 4.21: M/W SN general structure.

K-best MIMO detectors

In the K-best MIMO detectors detailed in [91, 96, 98–100], we observe that for 16-QAM and 64-QAM modulations ($Q = 16$ and $Q = 64$) at least 5-best and 10-best nodes ($W = 5$ and $W = 10$) are required respectively. Moreover, according to [91, 96, 99, 100] a typical value for the data width is sixteen bits, $N = 16$. So, for real-value detectors we have $M = W \cdot \sqrt{Q}$, namely $M = 20$ and $M = 80$ for 16-QAM and 64-QAM respectively.

Both the architectures proposed in [91] and [96] deal with a 4×4 64-QAM K-best MIMO detector. In particular, [91] relies on the bubble-sort approach whereas in [96] tree-sort is applied.

Non-binary LDPC decoder architectures

The non-binary LDPC decoder architectures proposed in [93,94] deal with codes in GF(32) and GF(64), that is $M = 32$ and $M = 64$ respectively. Moreover, they operate on a reduced number of messages, at least sixteen, that is $W = 16$ and we fix the data width to five bits ($N = 5$) as in [94].

The bubble check sorter proposed in [93] relies on a simplified extended min-sum (EMS) algorithm for check node processing that reduces the EMS original complexity from the order of W^2 to the order of $W\sqrt{W}$. In [101] it is implemented in several sequential rounds. On the other hand, in this current work, following the original description in [93] we implemented it as parallel architecture relying of a matrix structure. It is worth pointing out that, since the data in each row of the matrix described in [93] are supposed to be in order, our re-implementation of the bubble-check architecture has been equipped with a pre-sorting circuit. In [94], a reduced complexity sorter for the check node unit of a non-binary LDPC decoder is proposed. However, such an architecture relies on d_c rounds, where the M inputs are sliced and analyzed sequentially round by round. Since in this current work we deal with parallel sorting only, the architecture proposed in [94] is not considered in the comparison.

Turbo product code architectures

In the Chase-Pyndiah algorithm [83] a selection of the least reliable bits is necessary. As an example, in [95] a parallel implementation of turbo product codes that requires parallel partial sorting is addressed. Thus, in this section results for $M = 32{,}64$ and $W = 3{,}4$ are presented. The data width is five bits, $N = 5$, as in [95].

Comparisons

The BWA, FPCG and PS architectures, as well as the reference architectures in [74,91,93, 95–97] are all described in VHDL, simulated with ModelSim, synthesized using Synopsys Design Compiler, then placed and routed using Cadence SoC Encounter on a TSMC 90 nm CMOS standard-cell technology (where the area of a two-input NAND gate is 2.82 μm^2). Thanks to its scalability the BWA architecture can be easily adapted to the whole range of M, W and N values of the three considered applications. In Table 4.1 area (A) and critical path delay (C) for each architecture are compared. As it can be observed, the

Table 4.1: Post place and route results comparing area (A) [μm^2], critical path delay (C) [ns] and area-delay-product (P=A·C) [mm^2 · ns] for different architectures.

	K-best MIMO detectors		Non-binary LDPC decoder		Turbo product code	
	M, W, N 20, 5, 16 A C P	M, W, N 80, 10, 16 A C P	M, W, N 32, 16, 5 A C P	M, W, N 64, 16, 5 A C P	M, W, N 32, 3, 5 A C P	M, W, N 64, 4, 5 A C P
SN [74]	34043 10.6 0.36	254977 42.78 10.91	28932 23.36 0.68	65988 39.27 2.59	14597 6.24 0.09	33213 10.61 0.35
PB [97]	43286 10.8 0.47	305753 38.06 11.64	22068 7.31 0.16	55267 10.54 0.58	19507 6.35 0.12	52778 10.20 0.54
[91]	33484[(a)] 19.68[(a)] 0.66[(a)]	93171 40.98 3.82	- - -	- - -	- - -	- - -
[96]	62221[(a)] 15.13[(a)] 0.94[(a)]	406108 45.35 18.42	- - -	- - -	- - -	- - -
[93]	- - -	- - -	29735[(b)] 37.96[(b)] 1.13[(b)]	47157[(b)] 40.86[(b)] 1.93[(b)]	- - -	- - -
BWA	9345 16.92 0.16	29880 47.51 1.42	6153 36.29 0.22	9793 40.59 0.40	4562 6.72 0.03	8510 10.28 0.09
FPCG	67913 2.89 0.20	1532464 10.35 15.86	234952 7.50 1.76	943145 12.27 11.57	45815 2.21 0.10	349082 3.60 1.26
PS	28015 4.03 0.11	193024 11.32 2.19	44690 6.40 0.29	84655 9.45 0.80	8105 2.41 0.02	24892 3.64 0.09

[(a)] Obtained by extending the architecture to the test case.
[(b)] Obtained by adding a pre-sorting circuit.

proposed BWA architecture features the lowest complexity among the solutions compared in Table 4.1. The area of the BWA architecture is indeed less than half the area of the reference solutions and in the worst case it is about half the complexity of PS architecture. Besides, the critical path delay of the BWA architecture is almost comparable with that of other implementations. Moreover, if we compute the area-delay-product (P=A·C), the proposed BWA architecture is comparable with the PS architecture and is better than most of the other compared solutions. Further experiments adding pipelining have shown proportional throughput increase and an area overhead of BWA architectures always less than 35%. As for FPCG architecture, compared to the best reference architecture it has an overwhelming advantage on the speed which achieves 3.6 times faster on average and even 5.2 times faster at best, but its hardware cost is rather large. Finally the PS architecture extended from the work in [95] achieves a good balance on the timing performance and area, which has a comparable speed with FPCG architecture and a acceptable hardware cost.

Chapter 5

Conclusion

Currently full HD video prevails widely and ultra HD content is of growing popularity. Furthermore, according to visual networking index [102], by 2020, video content is expected to cause more than 75% of the global mobile data traffic. Whereas network bandwidth is still a valuable commodity. Consequently the efficiency of video coding has a significant impact on the transmission and storage costs. Thus developing video codecs of high performance and efficiency along with the newest video coding standard, HEVC, is of great meaning. In this thesis work, the main concerning is on the computational complexity reduction and improvement of timing performance of motion estimation algorithms for HEVC and the implementation and optimization of their VLSI architectures.

5.1 Contributions

1. A new SAD computing architecture is developed to alleviate the computational load of full search for ME algorithm. The results of simulation shows that the proposed architecture can reduce up to 70% of the computations with respect to the original FS algorithm with an imperceptible bit-rate increase and PSNR loss.

2. An enhanced TZ Search algorithm is explored with new strategy of multiple initial search centers and shared MVP on CU-level. It achieves a reduction of about 5% on the whole encoding time with negligible bit-rate increment and video quality degradation. Additionally the hardware implementation of the proposed architecture employs several parallelization mechanisms including parallel ME engines for PUs within a CU based on the shared MVP strategy, parallel initial searching centered at multiple MVPs, and the parallelized SAD calculating tree for PU size 32×32. Hence, with these parallelized components the proposed architecture is expected to obtain an appreciable improvement on throughput and encoding efficiency.

3. Three VLSI architectures are explored for finding the first W maximum/minimum values: the BWA architecture is based on the novel bit-wise AND scheme; the PS and FPCG architectures utilize the optimized comparator-based structure. Experimental results exhibit that, the BWA architecture has a very low hardware cost which is half of the area of the reference solutions in the worst case and its critical path delay is almost comparable with other implementations. As for FPCG architecture, compared to the reference architectures, it has an overwhelming advantage on the speed which achieves 3.6 times faster on average and even 5.2 times faster at best, but its hardware cost is rather large. Finally the PS architecture achieves a good balance on the timing performance and area, which has a slightly lower or comparable speed with FPCG architecture and a acceptable hardware cost.

5.2 Future Work

The enhanced TZ search algorithm has been simulated in the HEVC test model and hardware implementation is almost finished. In the following the verification, synthesis and physical design will be carried out to evaluate its hardware performance. Moreover the proposed new SAD computing structure can be combined in the enhanced TZ search architecture, with some optimization it is expected to further improve the coding efficiency. Since in the new SAD calculating strategy the sum of block of pixels is required, it can be applied in the embedded compression (EC) algorithm [103] for reducing memory bandwidth. Thus it will save the computations for summing of pixels which is also necessary in EC algorithm.

Bibliography

[1] G. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, Dec 2012.

[2] I. E. Richardson, *The H.264 Advanced Video Compression Standard, 2nd Edition*. Wiley, 2010, ISBN: 9780470516928.

[3] P. Helle, S. Oudin, B. Bross, D. Marpe, M. O. Bici, K. Ugur, J. Jung, G. Clare, and T. Wiegand, "Block merging for quadtree-based partitioning in hevc," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1720–1731, Dec 2012.

[4] A. V. Paramkusam and V. S. K. Reddy, "The efficient optimal and suboptimal motion estimation algorithms," *Signal, Image and Video Processing*, vol. 9, no. 6, pp. 1265–1270, Sep 2015.

[5] A. M. Tekalp, *Digital Video Processing*. Prentice Hall, 1995, ISBN: 0131900757.

[6] A. C. Bovik, *Handbook of Image and Video Processing*. Academic Press, 2000, ISBN: 0121197905.

[7] EBU, "High definition (hd) image formats for television production," *EBU TECH 3299*, pp. 1–10, 2010.

[8] S. Westland, "Models of the visual system and their application to image-quality assessment," in *10th Congress of the International Colour Association*, Mar 2005, pp. 8–13.

[9] R. Hoffman, *Data compression in digital Systems*. Chapman Hall, 1997.

[10] Y. Wang, J. Ostermann, and Y. Zhang, *Video Processing and Communications*. Prentice Hall, 2002, ISBN: 0130175471.

[11] A. Rosenfeld, *Picture Processing by Computer*. Academic Press, 1969.

[12] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards - Algoritms and Architectures*. Kluwer Academic Publishers, 1997.

[13] C. A. Poynton, *Digital Video and HDTV*. Morgan Kaufmann, 2003, ISBN: 1558607927.

[14] ITU-T, "Video codec for audiovisual services at p×64 kbits/s," in *ITU-T Recommendation H.261, Version 2*, 1993.

[15] ITU-T and ISO/IEC-JTC1, "Generic coding of moving pictures and associated audio information part 2: Video," in *ITU-T recommendation H.262 and ISO/IEC 13818-2 (MPEG-2)*, 1994.

[16] ITU-T, "Video coding for low bit rate communications," in *ITU-T Recommendation H.263, Version 2*, 1998.

[17] ISO/IEC, "Coding of audiovisual objects part 2: Visual," in *ISO/IEC 14496-2 (MPEG-4)*, 1999.

[18] ITU, "Advanced video coding for generic audiovisual services," in *ITU-T recommendation H.264*, Nov 2007.

[19] ——, "High efficiency video coding," in *ITU-T recommendation H.265*, Nov 2007.

[20] B. Bross, W. J. Han, G. J. Sullivan, J. R. Ohm, and T. Wiegand, "High efficiency video coding (hevc) text specification draft 9," in *document JCTVC-K1003, ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC)*, Oct 2012.

[21] JVT-G050, "Draft itu-t recommendation and final draft international standard of joint video specification," in *ITU-T Rec. H.264/ISO/IEC 14-thinsp, 496-10 AVC*, 2003.

[22] V. Sze, M. Budagavi, and G. J. Sullivan, *High Efficiency Video Coding (HEVC): Algorithms and Architectures*. Springer, 2014, ISBN: 9783319068947.

[23] J. L. Lin, Y. W. Chen, Y. P. Tsai, Y. W. Huang, and S. Lei, "Motion vector coding techniques for hevc," in *IEEE 13th International Workshop on Multimedia Signal Processing (MMSP)*, Oct 2011, pp. 1–6.

[24] N. Purnachand, L. Alves, and A. Navarro, "Fast motion estimation algorithm for hevc," in *IEEE International Conference on Consumer Electronics*, Sep 2012, pp. 34–37.

[25] M. E. Sinangil, A. P. Chandrakasan, V. Sze, and M. Zhou, "Hardware-aware motion estimation search algorithm development for high-efficiency video coding (hevc) standard," in *19th IEEE International Conference on Image Processing*, Oct 2012, pp. 1529–1532.

[26] ——, "Memory cost vs. coding efficiency trade-offs for hevc motion estimation engine," in *19th IEEE International Conference on Image Processing*, Oct 2012, pp. 1533–1536.

[27] Z. Chen, P. Zhou, and Y. He, "Fast motion estimation for jvt," in *Joint Video Team (JVT) of ISO/IEC MPEG ITU-T VCEG document JVTG016*, Mar 2013.

[28] X. Yi, J. Zhang, N. Ling, and W. Shang, "Improved and simplified fast motion estimation for jm," in *Doc. JVT-P021, 16th JVT meeting*, Jul 2005.

[29] A. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," in *Proceedings of the Visual Communication Image Processing*, Jan 2002, pp. 1069–1079.

[30] X. Xu and Y. He, "Comments on motion estimation algorithms in current jm software," in *Joint Video Team (JVT) of ISO/IEC MPEG ITU-T VCEG document JVT-Q089, 17th JVT meeting*, Oct 2005.

[31] Y. Ko, H. Kang, and S. Lee, "Adaptive search range motion estimation using neighboring motion vector differences," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 2, pp. 726–730, May 2011.

[32] W. Dai, O. Au, S. Li, L. Sun, and R. Zou, "Adaptive search range algorithm based on cauchy distribution," in *IEEE Visual Communications and Image Processing*, Nov 2012, pp. 1–5.

[33] X. Lu and C. Xiao, "A new strategy to predict the search range in h.264/avc," in *IEEE International Conference on Multimedia and Expo*, Jun 2009, pp. 21–24.

[34] Z. Chen, Q. Liu, T. Ikenaga, and S. Goto, "A motion vector difference based on self-incremental adaptive search range algorithm for variable block size motion estimation," in *15th IEEE International Conference on Image Processing*, Oct 2008, pp. 1988–1991.

[35] Z. Shi, W. Fernando, and A. Kondoz, "Adaptive direction search algorithms based on motion correlation for block motion estimation," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 3, pp. 1354–1361, May 2011.

[36] A. K. Z. Shi, W.A.C. Fernando, "An efficient fast motion estimation in h.264/avc by exploiting motion correlation character," in *IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, May 2012, pp. 298–302.

[37] C. M. Kuo, Y. H. Kuan, C. H. Hsieh, and Y. H. Lee, "A novel predictionbased directional asymmetric search algorithm for fast blockmatching motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 6, pp. 893–899, Jun 2009.

[38] X. Bao, D. Zhou, P. Liu, and S. Goto, "An advanced hierarchical motion estimation scheme with lossless frame recompression and earlylevel termination for beyond high-definition video coding," *IEEE Transactions on Multimedia*, vol. 14, no. 2, pp. 237–249, Apr 2012.

[39] M. Sarwer and Q. Wu, "Adaptive variable block-size early motion estimation termination algorithm for h.264/avc video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 8, pp. 1196–1201, Jun 2009.

[40] N. Purnachand, L. Alves, and A. Navarro, "Improvements to tz search motion estimation algorithm for multiview video coding," in *IEEE International Workshop*

on Systems Signals and Image Processing, Apr 2012, pp. 388–391.

[41] C. M. Kuo, Y. H. Kuan, C. H. Hsieh, and Y. H. Lee, "A new search algorithm based on multi-octagon-grid," in *the 2nd International Congress on Image and Signal Processing*, Oct 2009, pp. 1–5.

[42] K. H. Ng, L. M. Po, K. M. Wong, C. W. Ting, and K. W. Cheung, "A search patterns switching algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 5, pp. 753–759, May 2009.

[43] A. Tourapis, O. Au, and M. Liou, "Predictive motion vector field adaptive search technique (pmvfast): Enhancing block based motion estimation," in *SPIE Conference on Visual Communication and Image Processing*, Jan 2001, pp. 883–892.

[44] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *National Telecommunication Conference*, 1981, pp. C9.6.1–C9.6.5.

[45] S. Zhu and K. Ma, "A new diamond search algorithm for fast block matching motion estimation," in *International conference on information, communications and signal processing*, Sep 197, pp. 292–296.

[46] J. Tham, S. Ranganath, M. Ranganath, and A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 4, pp. 369–377, 1998.

[47] L. Torres and M. Kunt, *Video Coding: The Second Generation Approach.* Springer, 1996, ISBN: 9780792396802.

[48] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, Nov 1998.

[49] S. Metkar and S. Talbar, *Motion Estimation Techniques for Digital Video Coding.* Springer, 2013, ISBN: 9788132210962.

[50] D. Wackerly, W. Mendenhall, and R. L.Scheaffer, *Mathematical Statistics with Applications.* Thomson Higher Education, 2008, ISBN: 0495385085.

[51] E. Deza and M. M. Deza, *Encyclopedia of Distances.* Springer, 2009, ISBN: 9783662443415.

[52] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards including high efficiency video coding (hevc)," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1669–1684, Dec 2012.

[53] K. McCann, B. Bross, W. J. Han, I. K. K. K. Sugimoto, and G. J. Sullivan, "High efficiency video coding (hevc) test model 13 (hm 13) encoder description," in *Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-O1002*, Oct

2013.

[54] P. Hanhart, M. Rerabek, F. D. Simone, and T. Ebrahimi, "Subjective quality evaluation of the upcoming hevc video compression standard," in *Proc. SPIE. 8499, Applications of Digital Image Processing XXXV*, Oct 2012, p. 84990V.

[55] F. Bossen, "Common test conditions and software reference configurations," in *Joint Collaborative Team on Video Coding (JCT-VC), Document JCTVC-L1110*, Jan 2014.

[56] T. Wiegand and H. Schwarz, *Source Coding: Part I of Fundamentals of Source and Video Coding.* now publishers, 2011, ISBN: 9781601984081.

[57] K. Sayood, *Introduction to Data Compression.* Elsevier, 2005, ISBN: 9780126208627.

[58] ——, *Digital Image Processing.* Prentice Hall, 2008, ISBN: 9780131687288.

[59] J. H. Jeong, N. Parmar, and M. H. Sunwoo, "Enhanced test zone search algorithm with rotating pentagon search," in *International SoC Design Conference (ISOCC)*, Nov 2015, pp. 275–276.

[60] N. Parmar and M. H. Sunwoo, "Enhanced test zone search motion estimation algorithm for hevc," in *International SoC Design Conference (ISOCC)*, Nov 2014, pp. 260–261.

[61] X. Li, R. Wang, X. Cui, and W. Wang, "Context-adaptive fast motion estimation of hevc," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2015, pp. 2784–2787.

[62] D. T. Nghia, T. S. Kim, H. J. Lee, and S. I. Chae, "A modified tz search algorithm for parallel integer motion estimation in high efficiency video coding," in *19th International Conference on Systems, Signals and Image Processing (IWSSIP)*, Nov 2015, pp. 97–98.

[63] N. Purnachand, L. N. Alves, and A. Navarro, "Complexity reduction methods for fast motion estimation in hevc," *Signal Processing: Image Communication*, vol. 39, no. Part A, pp. 280–292, Nov 2015.

[64] A. Behnaz, S. Reza, and T. Manuchehr, "Optimized predictive zonal search (opzs) for block-based motion estimation," *Signal Processing: Image Communication*, vol. 39 Part A, pp. 293–304, Nov 2015.

[65] ITU-T and ISO/IEC-JTC-1, "Advanced video coding for generic audiovisual service," in *ITU-T and ISO/IEC JTC 1 Recommendation H.264 and ISO/IEC 14 496-10(MPEG-4) AVC*, 2003.

[66] K. McCann, "Samsung's response to the call for proposals on video compression technology," in *JCTVC A124, 1th JCT-VC Meeting*, Apr 2010, pp. 15–23.

[67] A. Fujibayashi, "Simplified motion vector prediction," in *JCT-VC D231, 4th Meeting*, Jan 2011, pp. 20–28.

[68] B. Bross, "Mv coding and skip/merge operations," in *JCT-VC E481, 5th Meeting*, Mar 2011, pp. 16–23.

[69] T. Sugio, "Parsing robustness for merge/amvp," in *JCT-VC F470, 6th Meeting*, Jul 2011, pp. 14–22.

[70] B. Bross, "High efficiency video coding (hevc) text specification draft 6," in *JCTVC-H1003, 8th JCT-VC Meeting*, Feb 2012, pp. 1–10.

[71] L. Zhao, X. Guo, S. Lei, S. Ma, D. Zhao, and W. Gao, "Non-ce9: Simplification of amvp," in *JCTVC-H0316, 8th JCT-VC Meeting*, Feb 2012, pp. 1–10.

[72] F. Bossen, "Common test conditions and software reference configurations," in *JCTVC-H1100, 8th JCT-VC Meeting*, Feb 2012, pp. 1–10.

[73] Q. Yu, L. Zhao, and S. Ma, "Parallel amvp candidate list construction for hevc," in *Visual Communications and Image Processing (VCIP)*, Nov 2012, pp. 1–6.

[74] D. E. Knuth, *The Art of Computer Programming.* Addison-Wesley, 1998, vol. 3 - Sorting and Searching.

[75] V. E. Alekseyev, "Sorting algorithms with minimum memory," *Kibernetica, 5*, pp. 99–103, 1969.

[76] S. Papaharalabos, P. T. Mathiopoulos, G. Masera, and M. Martina, "Novel non-recursive max* operator with reduced implementation complexity for turbo decoding," *IET Communications*, vol. 6, no. 7, pp. 702–707, Jul 2012.

[77] M. Martina, S. Papaharalabos, P. T. Mathiopoulos, and G. Masera, "Simplified Log-MAP algorithm for very low-complexity turbo decoder hardware architectures," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 3, pp. 531–537, Mar 2014.

[78] F. Guilloud, E. Boutillon, and J. L. Danger, "λ-min decoding algorithm of regular and irregular LDPC codes," in *International Symposium on Turbo Codes and Related Topics*, 2003, pp. 451–454.

[79] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X. Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1288–1299, Aug 2005.

[80] S. Zezza, S. Nooshabadi, and G. Masera, "A 2.63 Mbit/s VLSI implementation of SISO arithmetic decoders for high performance joint source channel codes," *IEEE Transactions on Circuits and Systems I*, vol. 60, no. 4, pp. 951–964, Apr 2013.

[81] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 3, pp. 491–503, Mar 2006.

[82] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over GF(q)," *IEEE Transactions On Communications*, vol. 55, no. 4, pp. 633–643, Apr 2007.

[83] R. M. Pyndiah, "Near-optimum decoding of product codes: block turbo codes," *IEEE Transactions On Communications*, vol. 46, no. 8, pp. 1003–1010, Aug 1998.

[84] K. Gunnam, G. Choi, and M. Yeary, "A parallel VLSI architecture for layered decoding for array LDPC codes," in *IEEE International Conference on VLSI Design*, 2007, pp. 738–743.

[85] D. Oh and K. K. Parhi, "Min-sum decoder architectures with reduced word length for LDPC codes," *IEEE Transactions on Circuits and Systems I*, vol. 57, no. 1, pp. 105–115, Jan 2010.

[86] C. Condo, M. Martina, and G. Masera, "VLSI implementation of a multi-mode turbo/LDPC decoder architecture," *IEEE Transactions on Circuits and Systems I*, vol. 60, no. 6, pp. 1441–1454, Jun 2013.

[87] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Transactions on Communications*, vol. 47, no. 5, pp. 673–680, May 1999.

[88] E. Li, D. Declercq, and K. Gunnam, "Trellis-based extended min-sum algorithm for non-binary LDPC codes and its hardware structure," *IEEE Transactions on Communications*, vol. 61, no. 7, pp. 2600–2611, Jul 2013.

[89] C. L. Wey, M. D. Shieh, and S. Y. Lin, "Algorithms of finding the first two minimum values and their hardware implementation," *IEEE Transactions on Circuits and Systems I*, vol. 55, no. 11, pp. 3430–3437, Dec 2008.

[90] L. G. Amarù, M. Martina, and G. Masera, "High speed architectures for finding the first two maximum/minimum values," *IEEE Transactions on VLSI systems*, vol. 20, no. 12, pp. 2342–2346, Dec 2012.

[91] M. Shabany and P. G. Gulak, "A 675 mbps, 4x4 64-QAM K-Best MIMO detector in 0.13 um CMOS," *IEEE Transactions on VLSI systems*, vol. 20, no. 1, pp. 135–147, Jan 2012.

[92] B. Wu and G. Masera, "Efficient VLSI implementation of soft-input soft-output fixed-complexity sphere decoder," *IET Communications*, vol. 6, no. 9, pp. 1111–1118, Sep 2012.

[93] E. Boutillon and L. Conde-Canencia, "Bubble check: a simplified algorithm for elementary check node processing in extended min-sum non-binary LDPC decoders," *IET Electronics Letters*, vol. 46, no. 9, pp. 633–634, Apr 2010.

[94] X. Zhang and F. Cai, "Reduced-complexity decoder architecture for non-binary LDPC codes," *IEEE Transactions on VLSI systems*, vol. 19, no. 7, pp. 1229–1238,

Jul 2011.

[95] C. Leroux, C. Jego, P. Adder, M. Jezequel, and D. Gupta, "A highly parallel turbo product code decoder without interleaving resource," in *IEEE Workshop on Signal Processing Systems*, 2008, pp. 1–6.

[96] P. Tsai, W. Chen, X. Lin, and M. Huang, "A 4*4 64-qam reduced-complexity k-best mimo detector up to 1.5gbps," in *IEEE International Symposium on Circuits and Systems*, 2010, pp. 3953–3956.

[97] K. Gunnam, G. Choi, W. Wang, and M. Yeary, "Parallel VLSI architecture for layered decoding," Texas A&M University, Tech. Rep., May 2007, available online at http://dropzone.tamu.edu/TechReports.

[98] Y. Sun and J. R. Cavallaro, "Low-complexity and high-performance soft MIMO detecion based on distributed M-algorithm through trellis-diagram," in *IEEE International Conference on Acoustics, Speach and Signal Processing*, 2010, pp. 3398–3401.

[99] D. Patel, V. Smolyakov, M. Shabany, and P. G. Gulak, "VLSI implementation of a WiMAX/LTE compliant low-complexity high-throughput soft-output K-Best MIMO detector," in *IEEE International Symposium on Circuits and Systems*, 2010, pp. 593–596.

[100] T. Kim and I. Park, "Small-area and low-energy K-Best MIMO detector using relaxed tree expansion and early forwarding," *IEEE Transactions on Circuits and Systems I*, vol. 57, no. 10, pp. 2753–2761, Oct 2010.

[101] E. Boutillon, L. Conde-Canencia, and A. Al-Ghouwayel, "Design of a GF(64)-LDPC decoder based on the EMS algorithm," *IEEE Transactions on Circuits and Systems I*, vol. 60, no. 10, pp. 2644–2656, Oct 2013.

[102] "Infographic: Cisco visual networking index global mobile data traffic forecast update (2015-2020)," Tech. Rep., Feb 2016, available Online: http://www.cisco.com/c/dam/en/us/solutions/service-provider/vni-service-adoption-forecast/index.html.

[103] J. Kim and C. M. Kyung, "A lossless embedded compression using significant bit truncation for hd video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 20, no. 6, pp. 848–860, Jun 2010.